GDAL-OGR, le couteau suisse du géomaticien

Author:

Yves JACOLIN

Date: décembre 2006

Table des matières

Chapitre I Introduction

A Historique des versions

- 2011-09 : rajout doc utils/ecwhdr
- 2011-08 : rajout section utils
- 2011-06 : passage en rst et fin mise à jour 1.8.0, début 1.9.0
- 2010-12 : mise à jour à la version 1.8.0
- 2009-10 : migration du site Softlibre vers le wiki de GeoRezo.
- 2009-02/2009-03 : mise à jour à la version 1.6.0 de GDAL-OGR, début de la mise à jour à la version 1.7.0.
- 2007-04-21 : Intégration dans le wiki.

B Contributeurs

1 Traduction

- Yves Jacolin
- Jérémie Garniaux

2 Rédacteurs

- Yves Jacolin (chapitre Python, FAQ)
- Marie Silvestre (chapitre Python)
- Ludovic Granjon (chapitre Python)

3 Relecture

- Even Rouault

C Installation

1 Binaires GDAL/OGR

Le projet GDAL ne produit pas de binaire régulier téléchargeable (exécutable) pour chaque version. Cependant, des efforts sont réalisés pour produire des binaires prêt à l'emploi.

1.a FWTools

Le binaire FWTools pour les systèmes Linux et Windows inclus l'ensemble des bibliothèques, commandes, la gestion de Python et la documentation de GDAL et d'autres choses encore (dont le visualiseur OpenEV). Des informations supplémentaires sont disponibles sur le site de FWTools

La dernière version de FWtools pour Windows, la version 2.4.7, date de la pre-version pre-1.6 de GDAL. Pour bénéficier de la dernière et meilleure version, vous pouvez vous référez aux autres builds binaire mentionnés dans la section Windows plus bas.

Les binaires FWTools pour Linux doivent fonctionner sur presque tous les systèmes Linux Intel moderne (leur production a débuté avec la série expérimentales de la série des 3.0.X).

1.b Linux

1.b.i Fedora

Fedora, depuis la version 7, inclus les binaires GDAL.

1.b.ii Entreprise Linux GIS (ELGIS)

L'effort ELGIS fournie des RPMs de différentes applications SIG pour Enterprise Linux et ses dérivés (RHEL, CentOS, Scientific Linux).

1.b.iii OpenSuSE

Les RPM de GDAL pour OpenSuSE avec d'autres outils géospatial sont publiés sur http://download.opensuse.org/repositories/Application:/Geo/

1.b.iv Debian

L'équipe SIG de Debian maintient des paquets GDAL pour Debian GNU/Linux.

1.b.v Ubuntu

Un nouveau UbuntuGIS est disponible sur launchpad d'UbuntuGIS et fournit des paquets à jour de GDAL pour Ubuntu Hardy, Karmic et Lucid. Pour plus d'informations, vous pouvez consulter la page UbuntuGIS.

1.c MacOS X

William Kyngesburye maintien un ensemble de builds pour GDAL et d'autres logiciels SIG sur http://www.kyngchaos.com/software:frameworks

1.d Windows

- Tamas Szekeres maintien un ensemble complet de paquetages binaires pour Win32 et Win64 (compilé avec VC2003/VC2005/VC2008/VC2010) disponible à l'endroit suivant : http://www.gisinternals.com/sdk/
  Ces paquets sont basés sur le développement actuel et les branches stables compilées à partir du SVN journalier de GDAL. Les paquets SDK correspondant sont également disponibles pour téléchargement à partir de cet endroit. Les paquets -devel sont basé sur la version en développement (1.9.0dev au moment de la rédaction), et les paquets -stable sont basés sur la dernière branche de la version stable (1.8 au moment de la rédaction).
- MapServer pour Windows (MS4W) est un installeur populaire qui contient GDAL, MapServer, et le serveur web Apache. Maintenu par Gateway Geomatics.
- Des binaires Windows via MinGW sont disponibles ici http://map.hut.fi/files/Geoinformatica/win32/
- Geoinformatica-yy-mm-dd.zip contient GDAL (habituellement une version de développement), Perl-GDAL, Perl, et beaucoup d'autres choses.
- Des exécuables minimalistes pour Windows sont disponibles ici : http://download.osgeo.org/gdal/win32/1.6/gdalwin32exe160.zip
- D'autres plugins seront ajoutés au même endroit (comme Oracle/OCI) : http://download.osgeo.org/gdal/win32/
- Des binaires avec plus de fonctionnalités pour Windows, dont python, proj et la gestion C# sont disponibles comme part du paquetage FWTools (date d'avant la version pre-1.6 de GDAL).

1.e OSGeo4W

OSGeo4W est une distribution de binaire d'un grand ensemble de logiciels open source geospatial pour les environnements Win32 (Windows XP, Vista, etc). OSGeo4W inclus GDAL/OGR, GRASS, MapServer, OpenEV, uDig, ainsi que d'autres paquetages (environ 70 à l'été 2008).

Chapitre II Les commandes GDAL

A Utilitaires GDAL

Les programmes suivants sont distribués avec GDAL :

- :ref:`gdal.gdal.gdalinfo` - renvoi des informations sur un fichier.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_translate` - Copie un fichier raster, avec la possibilité de contrôler son format de sortie.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdaladdo` - Ajoute une pré-visualisation à un fichier.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdalwarp` - Déforme une image dans un nouveau système de coordonnées.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdaltindex` - Construit un index de tuile raster pour MapServer.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdalbuildvrt` - Construire un VRT à partir d'une liste de jeu de données.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_contour` - Contour à partir d'un Modèle Numérique de Terrain.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdaldem` - Outils pour analiser et visualiser les MNT.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.rgb2pct` - Convertit une image en RVB 24bit vers une palette 8bit.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.pct2rgb` - Convertit une image d'une palette 8bit en une image RVB 24bit.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_merge` - Construit une mosaïque à partir d'une série d'images.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal2tiles` - Créer une structure de tuile TMS, un KML et un simple visualiseur web.

Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_rasterize` - Transforme des couches vecteurs en couche raster.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdaltransform` - Transforme des coordonnées.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.nearblack` - Convertie des bords proches du noir/blanc à leurs valeurs exactes.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_retile` - "Retiles" un ensemble de tuiles et/ou construit des niveaux de pyramide tuilée.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_grid` - Créer un raster à partir de données découpées.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_proximity` - Calcul une carte de proximité en raster.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_polygonize` - Génère des polygones à partir d'un raster.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_sieve` - Filtre sieve de raster.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal_fillnodata` - Interpole dans les régions *nodata*.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdallocationinfo` - Interroger un raster pour une localisation donnée.
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.gdal-config` - Obtient des informations pour compiler des logiciels qui utilisent GDAL.
  Unknown interpreted text role "ref".

1 Créer de nouveaux fichiers

Accéder à un fichier existant est assez facile. Il suffit d'indiquer le nom du fichier ou du jeu de données en paramètre dans la ligne de commande. Par contre, créer un fichier est plus compliqué. Il peut être nécessaire d'indiquer le format à générer, diverses options de création affectant la manière dont il sera créé et éventuellement un système de coordonnées à définir. Plusieurs de ces options sont prises en charge par les différents modules GDAL et sont introduites ici.

- `-of format` : Sélectionne le format du fichier à créer. Les formats sont précisés par leur nom court tel que GTiff (pour GeoTIFF) ou HFA (pour Herdas Imagine). La liste des codes des formats peut être listée par le drapeau `--formats`. Seuls les formats décrits comme “rw” (Lecture-écriture) peuvent être créés.
  Plusieurs utilitaires créent par défaut des fichiers GéoTiff, si aucun format n'est spécifié. Les extensions des fichiers ne sont pas utilisées pour déterminer le format de sortie, ni ajoutées par GDAL si l'utilisateur les a omis.
- `-co NAME=VALUE` : Plusieurs formats ont une ou plusieurs options spécifiques de création qui peuvent être utilisé pour contrôler la création de fichier. Par exemple, le pilote GéoTiff supporte des options de créations pour préciser la compression, ou si le fichier doit être tuilé.
  Ces options de création disponibles varient en fonction du pilote et certains formats en n'ont pas du tout. La liste des options supportées pour chaque format peut être affichée avec le drapeau `--format <format>` en ligne de commande, mais la documentation Web du format est certainement l'endroit le plus approprié

pour obtenir tous les détails nécessaires.

- `-a_srs SRS` : Plusieurs utilitaires (dont `gdal_translate` et `gdalwarp`) incluent la possibilité de définir les systèmes de coordonnées dans la ligne de commande avec les options `-a_srs` (définit SRS à la sortie), `-s_srs` (source SRS) et `-t_srs` (cible SRS).
  Ces options permettent de définir le système de coordonnées (*SRS* signifie *Spatial Reference System*, système de référence spatial) de nombreuses façons :
  - **NAD27/NAD83/WGS84/WGS72 :** Ces systèmes de coordonnées géographiques (lat/lon) communs sont précisés directement avec ces termes.
  - **EPSG:n :** les Systèmes de coordonnées (projetés ou géographiques) sont indiqués en se basant sur leur code EPSG, par exemple EPSG:2154 correspond au Lambert 93 en France. Une liste des systèmes de coordonnées est disponible dans les fichiers gcs.csv et pcs.csv de GDAL.
  - **PROJ.4 Définitions :** Une chaîne de définition PROJ.4 est utilisée comme système de coordonnées. Par exemple, "+proj=utm +zone=11 +datum=WGS84". Prenez soin de garder la chaîne proj.4 en un morceau, sous la forme d'un seul argument, en utilisant des guillemets doubles.
  - **OpenGIS Well Known Text :** L'Open GIS Consortium a défini un format textuel pour décrire les systèmes de coordonnées comme une partie des spécifications Simple Features. Ce format est celui que GDAL utilise comme système de coordonnées. Le nom du fichier contenant une définition du système de coordonnées WKT, peut être utilisé comme argument du système de coordonnées, ou le système de coordonnées complet peut être utilisé dans la ligne de commande (bien que gérer tous les guillemets puisse être un défi).
  - **ESRI Well Known Text :** ESRI utilise une légère variation du format WKT de l'OGC dans leur produit ArcGIS (fichiers .prj), et ceux-ci peuvent être utilisés de la même manière que les fichiers WKT, mais le nom du fichier doit être précédé de *ESRI::*. Par exemple "ESRI::NAD 1927 StatePlane Wyoming West FIPS 4904.prj".
  - **Références spatiales à partir d'URL :** par exemple http://spatialreference.org/ref/user/north-pacific-albers-conic-equal-area/.
  - **Un nom de fichier :** Le nom d'un fichier contenant des définitions de système de coordonnées en WKT, chaînes PROJ.4, ou XML/GML peut être fourni.

2 Options de la ligne de commande

Tous les programmes en ligne de commande de GDAL supportent les options générales suivantes :

- `--version` : affiche la version de GDAL et termine.
- `--formats` : liste tous les formats raster supportés par cette compilation de GDAL (en lecture seule et en lecture écriture) et se termine. La gestion du format est indiquée comme suit :
  - 'ro' et un pilote en lecture seule ;
  - 'rw' est lu ou écrit (c'est à dire géré par *CreateCopy*);
  - 'rw+' est lu, écrit et mis à jour (c'est à dire géré par *Create*). Le caractère 'v' est ajouté pour les formats gérant l'IO virtuel (/vsimem, /vsigzip, /vsizip, etc). Note : Les formats valides pour la sortie de `gdalwarp` sont les formats qui gère la méthode *Create()* (marqué rw+), a seulement la méthode *CreateCopy()*.
- `--format format` : liste des informations détaillées sur le pilote du format. Le format doit être le nom court affiché par l'option --formats, tels que GTiff.

- `--optfile file` : lit le nom du fichier et substitut son contenu dans la liste des options de la ligne de commande. Les lignes débutantes par # sont ignorées. Des arguments de plusieurs mots peuvent être réunis en les entourant de guillemets.
- `--config key value` : définit la valeur de la clé dans la configuration, par opposition à la déclaration des variables d'environnements. Il existe des mots-clés de configuration communs tels que GDAL_CACHEMAX (mémoire utilisé en interne pour mettre en cache en mégaoctets) et GDAL_DATA (chemin du répertoire des "données" de GDAL). Des pilotes individuels peuvent être influencés par d'autres options de configuration.
- `--debug value` : contrôle quel message de débogage sera affiché. Une valeur à ON permettra l'affichage des messages de débogage. Une valeur à OFF n'affichera pas les messages de débogage. Une autre valeur sélectionnera seulement les messages de débogages contenant cette chaîne dans le code le précédent.
- `--help-general` : donne un bref message des usages des options en ligne de commande et termine.

B gdalinfo

Listes des informations sur le jeu de données raster.

Usage :

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

Le programme gdalinfo liste diverses informations sur les jeux de données raster supportés par GDAL.

- **-mm :** Force le calcul des valeurs min/max réelles pour chaque bande dans le jeu de données.
- **-stats :** Lit et affiche des statistiques de l'image. Force le calcul si aucune n'est présente dans l'image.
- **-hist :** Renvoie des informations sur l'histogramme pour toutes les bandes.
- **-nogcp :** Supprime l'affichage de la liste des points d'amer. Il peut être utile pour certains jeux de données avec un grand nombre de points d'amer (GCP) tel que L1B AVHRR ou HDF4 MODIS qui en contiennent des centaines.
- **-nomd :** Supprime l'affichage des méta-données. Certains jeux de données peuvent contenir beaucoup de méta-données.
- **-noct :** Supprime l'impression des tables de couleurs.
- **-checksum :** Force le calcul d'un checksum pour chaque bande du jeu de données.
- **-mdd domaine :** Affiche les méta-données pour le domaine défini.

Le programme ''gdalinfo'' affichera les données suivantes (si elles sont connues) :

- Le pilote du format utilisé pour lire le fichier.
- La taille du raster (en pixels et en lignes).
- Le système de coordonnées du fichier (au format OGC WKT).
- La géotransformation associée avec le fichier (les coefficients de rotation ne sont généralement pas affichés).
- Les coordonnées des bornes dans le système géoréférencé et si possible basé sur lat/long sur la géotransformation entière (mais pas les points d'amer ou GCP).

- Les points d'amer (GCP en anglais).
- Les méta-données des gros fichiers (incluant les sous-jeux de données).
- Les types de données des bandes.
- Les interprétations de couleurs des bandes.
- La taille des blocs des bandes.
- Les descriptions des bandes.
- Les valeurs min/max des bandes (connues en interne et calculables).
- Checksum des bandes (si le calcul est demandé).
- La valeur NODATA des bandes.
- Les résolutions des aperçus des bandes disponibles.
- Le type d'unité des bandes (c'est à dire "mètres" ou "pied" pour les bandes d'élévation).
- Les tables de pseudo-couleurs des bandes.

Exemple :

```
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
```

C gdal_translate

Converti des données raster en différents formats.

Usage :

```
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"]*
   src_dataset dst_dataset
```

L'utilitaire `gdal_translate` peut être utilisé pour convertir des données raster en différents formats, et éventuellement réaliser des opérations comme re-échantillonner, réduire les pixels pendant le calcul.

- **-ot : type** Pour que la bande en sortie soit du type de données indiqué.
- **-strict :** N'oublie pas les pertes de données et les erreurs lors de la transformation vers le format de sortie.
- **-of format :** Sélectionne le format de sortie. Celui par défaut est le GéoTiff (GTiff). Utilisez les noms de formats courts.
- **-b band :** Sélectionne une bande en entrée pour la sortie. Les bandes sont numérotées à partir de 1. De multiples options `-b` peuvent être utilisées pour sélectionner une série de bandes en entrée pour l'insertion dans le fichier de sortie ou pour ordonner les bandes.
- **-mask *band* :** (GDAL >= 1.8.0) Sélectionne une bande *band* en entrée pour créer une bande de masquage dans le jeu de données en sortie. Les bandes sont numéroté à partir de 1. *band* peut être définie à "none" pour éviter de copier le masque global du jeu de données en entrée s'il existe. Autrement il sera copié par défaut ("auto"), sauf si le masque est un canal alpha ou s'il est explicitement utilisé pour une bande normale du jeu de données en sortie ("-b mask"). *band* peut également être définie à "mask,1" (ou seulement "mask") pour signifier la bande de masque de la première bande du jeu de données en entrée.
- **-expand rgb|rgba:** (From GDAL 1.6.0) Présente un jeu de donné avec une bande munie d'une table de couleur comme un fichier avec 3 (RVB) ou 4 (RVBA) bandes. Utile pour des pilotes comme JPEG, JPEG2000, MrSID, ECW qui ne gèrent pas des rasters avec les tables de couleurs. La valeur 'gray' (à partir de GDAL 1.7.0) permet d'étendre un jeu de données avec une table de couleur qui contient seulement les niveaux de gris en un jeu de données de gris indéxé.
- **-outsize xsize[%] ysize[%] :** Définit la taille du fichier exporté. L'unité est le pixel et ligne sauf si '%' est utilisé auquel cas il est une fraction de la taille de l'image en entrée.
- **-scale [src_min src_max [dst_min dst_max]] :** Redimensionne la valeur du pixel en entrée à partir du domaine *src_min* à *src_max* et du domaine *dst_min* à *dst_max*. S'il est omis le domaine de sortie est de 0 à 255. S'il est omis le domaine en entrée est automatiquement calculé à partir des données sources.
- **-unscale :** Applique la méta-donnée d'échelle/d'offset pour les bandes à convertir de valeurs ajustées à des valeurs non ajustées. Il est aussi souvent nécessaire de redéfinir le type de données en sortie avec le paramètre **-ot**.
- **-srcwin xoff yoff xsize ysize :** Sélectionne une région à partir de l'image source pour copie basée sur la location pixel/ligne.
- **-projwin ulx uly lrx lry :** Sélectionne une région à partir de l'image source pour

copie (comme *-srcwin*) mais avec les bords en coordonnées géoréférencées.

- **-a_srs srs_def :** Écrase la projection du fichier de sortie. La paramètre *srs_def* peut être de la forme de n'importe quel de ceux acceptés par GDAL/ORG, WKT, PROJ4, EPSG:n ou un fichier contenant un WKT.
- **-a_ullr ulx uly lrx lry :** Assigne/écrase les bornes géoréférencées du fichier géoréférencé. Cela définit les bornes géoréférencées au fichier de sortie, en ignorant les conséquences.
- **-a_nodata value :** Assigne une valeur nodata définit aux bandes exportées. À partir de GDAL 1.8.0, ce paramètre peut être définie à *none* pour éviter de définir une valeur nodata dans le fichier en sortie si elle existe pour le fichier source.
- **-mo "META-TAG=VALUE" :** Définir une clé et une valeur d'une méta-données aux données en sortie si possible.
- **-co "NAME=VALUE" :** Définit une option créée pour le pilote du format de sortie. Des options -co multiples peuvent être listées. Voyez la documentation du format définit pour la création d'options pour chaque format.
- **-gcp pixel line easting northing elevation :** Ajoute un point d'amer (GCP en anglais) indiqué aux données en sortie. Cette option peut être définie plusieurs fois pour fournir un ensemble de points d'amer (GCP).
- **-q :** Supprime l'affichage du suivi de progression et autre affichage d'informations.
- **-sds :** Copie tous les sous-ensembles de données de ce fichier en fichier indépendant. Utilisé avec les formats HDF ou OGDI qui possèdent des sous-ensembles de données.
- **src_dataset :** Le nom du fichier source. Il peut être soit un nom de fichier, une URL d'une source de données ou un nom d'un sous jeu de données pour les fichiers de plusieurs jeux de données.
- **dst_dataset :** Le nom du fichier de destination.

Exemple :

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

À partir de GDAL 1.8.0, pour créer un fichier TIFF avec une compression en JPEG avec un masque interne à partir un jeu de données RGBA :

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

À partir de GDAL 1.8.0, pour créer un jeu de données RGBA à partir d'un jeu de données RGB avec un masque :

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

D gdaladdo

Construit ou reconstruit les images d'aperçus.

Usage :

```
gdaladdo [-r
{nearest,average,gauss,cubic,average_mp,average_magphase,mode}]
     [-ro] [-clean] [--help-general] filename levels
```

Le programme `gdaladdo` peut être utilisé pour construire ou reconstruire les images d'aperçu pour la plupart des formats supportés avec un ou plusieurs algorithmes.

- **-r {nearest (default),average,gauss,cubic,average_mp,average_magphase,mode} :** Sélectionne un algorithme de re-échantillonnage.
- **-ro :** (Disponible à partir de gdal 1.6.0) Ouvre le jeu de données en mode lecture seul, dans le but de générer un aperçu externe (notamment pour les GeoTIFF).
- **-clean :** (disponible à partir de GDAL 1.7.0) enlève les aperçues.
- **filename :** Le fichier pour lequel on veut construire les aperçus (ou dont les aperçues doivent être enlevés).
- **levels :** Une liste de niveaux d'aperçu à construire. Ignoré avec l'option *-clean*.

Le *mode* (disponible à partir de gdal 1.6.0) sélectionne la valeur de tous les points de l'échantillon qui apparaît le plus souvent. *average_mp* n'est pas utilisable. *Average_magphase* donne une moyenne des données complexes dans l'espace mag/phase. *Nearest* et *average* sont applicable aux données des images normales. *Nearest* applique un re-échantillonnage du plus proche voisin (échantillonnage simple), tandis que *average* calcul la moyenne de tous les pixels différents de NODATA. Le re-échantillonnage *Cubic* (disponible à partir de GDAL 1.7.0) applique un motif de convolution cubique approximativement de 4x4. *Gauss resampling* (disponible à partir de gdal 1.6.0) applique une transformation de Gaussian avant le calcul de l'aperçu ce qui peut améliorer les résultats par rapport à une moyenne simple dans le cas des bords nets avec un fort contraste ou avec du bruit. Les valeurs des niveaux conseillés devraient être 2, 4, 8 ... afin qu'un ré-échantillonnage Guassien de 3x3 soit sélectionné.

`gdaladdo` honorera correctement les tuples *NODATA_VALUES* (méta-donnée spéciale du jeu de données) afin que seul un triplet RGB donné (dans le cas d'une image RGB) sera considéré comme valeur *nodata* et pas chaque valeur du triplet indépendamment pour chaque bande.

La sélection d'une valeur de niveau 2 entraîne le calcul d'un niveau d'aperçu de la moitié de la résolution (pour chaque dimension) de la couche de base. Si le fichier possède des niveaux de résolution au niveau sélectionné, ces niveaux seront recalculés et écrits de nouveau.

Certains pilotes de format ne supportent pas les aperçus. Plusieurs pilotes de formats rangent les aperçus dans un fichier secondaire avec l'extension .ovr qui est en réalité un format tiff. Par défaut le pilote GeoTIFF range les aperçus à l'intérieure du fichier utilisé (s'il peut être écrasé), sauf si l'option `-ro` est définie.

La plupart des pilotes gère également un format d'aperçu alternatif en utilisant le format Erdas Imagine. Pour déclencher ceci, utilisez l'option de configuration *USE_RRD=YES*. Cela placera les aperçus dans un fichier .aux associé disponible pour une utilisation directe avec Imagine ou ArcGIS ainsi que les applications GDAL (par exemple --config USE_RRD YES).

1 Aperçus externes dans le format GeoTIFF

Les aperçus externes créés dans un format TIFF peuvent être compressés en utilisant l'option de configuration `COMPREES_OVERVIEW`. Toutes les méthodes de compression,

supportées par le pilote GeoTIFF, sont disponibles ici (par exemple `--config COMPRESS_OVERVIEW DEFLATE)`. L'interprétation photométrique peut être définie avec `--config PHOTOMETRIC_OVERVIEW {RGB,YCBCR,...} ``, et l'entrelacement avec ``--config INTERLEAVE_OVERVIEW {PIXEL|BAND}`.

Pour les aperçus externes compressés en JPEG, la qualité JPEG peut être définie avec "--config JPEG_QUALITY_OVERVIEW value" (GDAL 1.7.0 ou plus récent).

Pour les aperçus externes compressés en LZW ou DEFLATE, la valeur prédite peut être définie avec "--config PREDICTOR_OVERVIEW 1|2|3" (GDAL 1.8.0 ou plus récent).

Pour produire des aperçus en JPEG dans TIFF le plus petit possible, vous devez utiliser :

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

À partir de GDAL 1.7.0, les aperçues externes peuvent être créé au format BigTIFF en utilisant l'option de configuration `BIGTIFF_OVERVIEW` : `--config BIGTIFF_OVERVIEW {IF_NEEDED|IF_SAFER|YES|NO}`. La valeur par défaut est `IF_NEEDED`. Le comportement de cette option est exactement le même que l'option de création `BIGTIFF` documentée dans la documentation du pilote :ref:`gdal.gdal.formats.gtiff`.

Unknown interpreted text role "ref".

- YES force BigTIFF.
- NO force des TIFF classiques.
- IF_NEEDED créera seulement un BigTIFF si cela est clairement nécessaire (non compressé, et des aperçues plus grande que 4Go).
- IF_SAFER créera un BigTIFF si le fichier résultant *pourrait* excéder 4Go.

Voyez la documentation du pilote GeoTIFF pour des explications supplémentaires sur toutes ces options.

Exemple :

Créer des aperçus, inclus dans le fichier TIFF fournit :

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

Créer un fichier d'aperçu externe en GeoTIFF compressé à partir du fichier ERDAS .IMG :

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

Créer un fichier d'aperçu du GeoTIFF compressé en JPEG à partir d'un jeu de données RVB à 3 bandes (si le jeu de données est un GeoTiff que l'on peut écrire, Vous avez également besoin d'ajouter l'option *-ro* pour forcer la génération des aperçues externes) :

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

Créer des aperçus au format Erdas Imagine pour le fichier JPEG indiqué :

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

E gdalwarp

Utilitaire de déformation et de re-projection d'image simple.

Usage :

```
gdalwarp [--help-general] [--formats]
    [-s_srs srs_def] [-t_srs srs_def] [-to "NAME=VALUE"]
    [-order n] [-tps] [-rpc] [-geoloc] [-et err_threshold]
    [-te xmin ymin xmax ymax] [-tr xres yres] [-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]
    [-of format] [-co "NAME=VALUE"]*
    srcfile* dstfile
```

La commande `gdalwarp` est une commande de déformation et de re-projection d'image. Le programme peut re-projeter dans n'importe quelles projections supportées, et appliquer les points d'amer stockés avec l'image si l'image est « brute » avec les informations de contrôle.

- **-s_srs srs def :** Définition de la référence spatiale de la source. Le système de coordonnées qui peut être utilisé sont ceux supportés par l'appel de OGR Spatial *Reference.SetFromUserInput()*, ce qui inclut les déclarations EPSG, GCS (c'est-à-dire EPSG:4296), PROJ.4 (comme ci-dessus), ou le nom d'un fichier .prf contenant un texte « well known ».
- **-t_srs srs_def :** Définition de la référence spatiale de la cible. Le système de coordonnées qui peut être utilisé sont ceux supportés par l'appel de OGR Spatial *Reference.SetFromUserInput()*, ce qui inclut les déclarations EPSG, GCS (c'est-à-dire EPSG:4296), PROJ.4 (comme ci-dessus), ou le nom d'un fichier .prf contenant un texte « well known ».
- **-to NAME=VALUE :** définie une option de transformation disponible pour être envoyé à la méthode *DALCreateGenImgProjTransformer2()*.
- **-order n :** ordre de l'équation polynomiale utilisé pour déformer (1 à 3). La valeur par défaut pour choisir un ordre polynomial se base sur le nombre de points d'amer.
- **-tps :** active l'utilisation de la transformation de type « thin plate spline » basé sur les points d'amer disponible.
- **-rpc :** force l'utilisation des RPC.
- **-geoloc :** force l'utilisation des tableaux Geolocation.
- **-et err_threshold :** seuil d'erreur pour l'approximation de la transformation (en pixel – par défaut à 0,125).
- **-te xmin ymin xmax ymax :** définie l'étendue géoréférencée du fichier à créer en sortie.
- **-tr xres yres :** définie la résolution du fichier en sortie (en unité du géoréférencement cible).
- **-ts width height :** définie la taille du fichier en sortie en pixels et lignes. si *width* ou *height* est définie à 0, les autres dimensions seront supposées à partir de la résolution calculée. Notez que `-ts` ne peut pas être utilisé avec `-tr`.
- **-wo "NAME=VALUE" :** définie une option de déformation. La doc de *GDALWarpOptions::papszWarpOptions* montre toutes les options. Multiple

options **-wo** peuvent être listé.

- **-ot type :** pour définir le type des bandes en sortie.
- **-wt type :** type de donnée des pixels de travail. Les types de données des pixels dans l'image source et les buffers des images de destinations.
- **-r :** Méthode de rééchantillonnage à utiliser. Les méthodes disponibles sont :
    - *near* : interpolation du plus proche voisin (par défaut, l'interpolation la plus rapide, mais de moins bonne qualité).
    - *bilinear* : interpolation bilinéaire.
    - *cubic* : interpolation cubique.
    - *cubicspline* : interpolation cubique spline (algorithme le plus lent).
    - *lanczos* : interpolation *Lanczos windowed sinc*.
- **-srcnodata value [value...] :** définie les valeurs de masques nodata pour les bandes en entrées (différentes valeurs peuvent être indiquées pour chaque bande). Si plus d'une valeur est indiquée, toutes les valeurs devront être entourées de guillemets pour être considérées comme un seul argument. Les valeurs masquées ne seront pas utilisées pour l'interpolation. Utilisez une valeur `None` pour ignorer les définitions de nodata intrinsèque du jeu de données sources.
- **-dstnodata value [value...] :** définie les valeurs de masques nodata pour les bandes en sortie (différentes valeurs peuvent être indiquées pour chaque bande). Si plus d'une valeur est indiquée toutes les valeurs devront être entourées de guillemets pour être considérées comme un seul argument. Les nouveaux fichiers seront initialisés avec cette valeur et si possible la valeur nodata sera enregistrée dans le fichier en sortie.
- **-dstalpha :** Créé une bande alpha en sortie pour identifier les pixels nodata (non définie/transparent).
- **-wm memory_in_mb :** définie la quantité de mémoire (en mégaoctets) que l'API de déformation est autorisée à utiliser pour la mise en cache.
- **-multi :** utilise une implémentation multithread pour la déformation. De multiples threads seront utilisés pour calculer les morceaux de l'image et exécuter des opérations d'entrée/sortie simultanément.
- **-q :** mode silencieux.
- **-of format :** sélectionne le format de sortie. Par défaut, GéoTIFF (Gtiff). Utilisez le format de nom court.
- **-co "NAME=VALUE" :** passe une option de création au pilote du format en sortie. De multiples options `-co` peuvent être listées. Lisez la documentation spécifique du format pour les options de création légales pour chaque format.
- **-cutline datasource :** active l'utilisation d'une ligne de découpage flou à partir du nom de la source de données OGR.
- **-cl layername :** sélectionne la couche nommée à partir du jeu de données de lignes de découpage.
- **-cwhere expression :** restreint des objets des lignes de découpage désirées basé sur une requête attributaire.
- **-csql query :** sélectionne les objets des lignes de découpage en utilisant une requête SQL au lieu de se baser sur une couche avec l'option `-cl`.
- **-cblend distance :** définie une distance floue à utiliser pour flouter les lignes de découpage (en pixels).
- **srcfile :** le nom du fichier source.
- **dstfile :** le nom du fichier de destination.

Le mosaïquage dans un fichier en sortie qui existe déjà est supporté si le fichier existe déjà. L'étendue spatiale d'un fichier existant ne sera pas modifiée pour s'accommoder

aux nouvelles données, vous pouvez donc l'enlever dans ce cas.

Des lignes de découpage polygonales peuvent être utilisées pour restreindre la zone du fichier de destination qui peut être mis à jour, incluant leur mélange. Les éléments des lignes de découpage doivent être présents dans les unités géographiques des fichiers de destination.

Exemple :

Par exemple, une scène spot de 8 bits stocké dans un fichier GéoTIFF avec les points d'amer aux coins en lat/long peuvent être déformé en une projection UTM avec la commande suivante :

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

Par exemple, le second canal d'une image RASTER stocké au format HDF avec les points d'amer des coins en lat/long peut être déformé en une projection UTM avec la commande suivante :

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

F gdaltindex

Construit un index de tuile de raster dans un shapefile.

Usage :

```
gdaltindex [-tileindex field_name] [-write_absolute_path] \
  [-skip_different_projection] index_file [gdal_file]*
```

Ce programme construit un shapefile avec un enregistrement pour chaque fichier raster en entré, un attribut contenant le nom du fichier, et un objet polygone entourant le raster. Ce fichier est utilisable dans MapServer comme une mosaïque de raster (tileindex).

- Le shapefile (index_file) sera créé s'il n'existe pas, autrement il sera ajouté à celui existant.
- Le champ d'indexation des tuiles par défaut est 'location'.
- Les noms des fichiers raster seront inclus dans le fichier exactement comme ils sont dénommés dans la ligne de commande à moins que l'option *-write_absolute_path* soit utilisée.
- Si *-skip_different_projection* est définie, seuls les fichiers avec le même fichier de référence de projection que ceux dans le tileindex seront insérés.
- Des polygones rectangulaires simples sont générés dans le même système de coordonnées que les rasters.

Exemple :

```
gdaltindex doq_index.shp doq.tif
```

G gdalbuildvrt

Construit un VRT à partir d'une liste de jeu de données.

1 Usage

```
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]*
```

2 Description

Ce programme construit un fichier VRT (Jeu de données virtuel) qui est une mosaïque d'une liste de jeux de données de GDAL en entrée. La liste des jeux de données de GDAL en entrée peut être définie à la fin de la ligne de commande, ou placer dans un fichier texte (un nom de fichier par ligne) pour les listes importantes, ou cela peut être un index de tuile de MapServer (voir l'utilitaire :ref:`gdal.gdal.gdaltindex`). Dans ce dernier cas, toutes les entrées dans l'index de tuile seront ajoutées au VRT.

Unknown interpreted text role "ref".

Avec l'option *-separate*, chaque fichiers est placé dans une bande séparée et empilée dans les bandes VRT. Autrement, les fichiers sont considéré comme des tuiles d'une mosaïque plus importante et le fichier VRT a autant de bande que ceux des fichiers en entrée.

Si un jeu de données de GDAL est réalisé à partir de plusieurs sous jeu de données et ne possède aucune bande raster, tous les sous jeux de données seront ajoutés au fichier VRT plutôt que le jeu de données lui-même.

`gdalbuildvrt` réalise quelques vérifications pour s'assurer que tous les fichiers qui seront mis dans le VRT ont des caractéristiques similaires : nombre de bandes, projection, interprétation de couleur... Sinon, les fichiers qui ne correspondent pas aux caractéristiques communes seront ignorés.

S'il y a un certain taux de recouvrement, ente les fichiers, l'ordre de superposition peut dépendre de l'ordre où ils sont insérés dans le fichier VRT, mais vous ne devez pas compter sur ce comportement.

Cette commande est en partie différente de la commande `gdal_vrtmerge.py` et est compilé par défaut à partir de GDAL 1.6.1.

- **-tileindex :** utilise la valeur définie comme champ d'index de tuile, au lieu de la valeur par défaut, 'location'.
- **-resolution {highest|lowest|average|user} :** Si la résolution de tous les fichiers en entrée n'est pas la même, l'option `-resolution` permet à l'utilisateur de contrôler la manière dont la résolution en sortie sera calculée. `average` est celle par défaut. 'highest' prendra les plus petites valeurs de la dimension des pixels au sein du jeu de raster. 'lowest' prendra les plus grandes valeurs de la

dimension des pixels au sein du jeu de raster. 'average' calculera une moyenne des dimensions des pixels au sein du jeu de rasters. 'user' est nouveau dans GDAL 1.7.0 et doit être utilisé en combinaison avec l'option *-tr* pour définir une résolution cible.

- **-tr xres yres :** (à partir de GDAL 1.7.0) définie la résolution cible. Les valeurs doivent être exprimées en unité géorérérencée. Les deux valeurs doivent être positives. Définir ces valeurs est bien sur incompatible avec les valeurs *highest*|*lowest*|*average* de l'option *-resolution*.
- **-tap :** (GDAL >= 1.8.0) (*target aligned pixels*) aligne les coordonnées de l'étendue du fichier en sortie avec les valeurs de l'option *-tr*, de telle sorte que l'étendue alignée inclus l'étendue minimale.
- **-te xmin ymin xmax ymax :** (à partir de GDAL 1.7.0) définie les étendues géoréférencées d'un fichier vrt. Les valeurs doivent être exprimées en unité du géoréférencement. Si celle-ci n'est pas définie, l'étendue du VRT est la boîte minimale du jeu des rasters sources.
- **-addalpha :** (à partir de GDAL 1.7.0) ajoute une bande de masque alpha au VRT quand le raster source en possède une. Utile principalement pour les sources RGB (ou les sources en niveau de gris). La bande alpha est rempli à la volée avec la valeur 0 dans les zones sans raster source, et la valeur 255 dans les zones avec un raster source. Cela permet à un visualiseur RGBA d'afficher les zones sans rasters source en transparence et les zones avec des rasters sources en opaque. Cette option n'est pas compatible avec l'option *-separate*.
- **-hidenodata :** (à partir de GDAL 1.7.0) même si les bandes contiennent des valeurs *nodata*, cette option permet au bande VRT de ne pas renvoyer la valeur *NoData*. Cela est utile lorsque vous désirez contrôler la couleur d'arrière plan du jeu de données. En l'utilisant en parallèle avec l'option *-addalpha*, vous pouvez préparer un jeu de données qui ne renvoie pas la valeur *nodata* mais est transparent dans la zone qui contient aucune donnée.
- **-srcnodata value [value...] :** (à partir de GDAL 1.7.0) définie les valeurs *nodata* au niveau de la bande en entrée (différentes valeurs peuvent être fournies pour chaque bande). Si plus d'une valeur est fournie toutes les valeurs doivent être entre guillemet pour les réunir ensemble comme argument unique. Si l'option n'est pas définie, les définitions *nodata* intrinsèques sur le premier jeu de donnée seron utilisé (s'ils existent). La valeur définie par cette option est écrit dans l'élément *NODATA* de chaque élément *ComplexSource*. Utilisez une valeur à *None* pour ignorer les définitions *nodata* intrinsèques des jeux de données sources.
- **-vrtnodata value [value...] :** (à partir de GDAL 1.7.0) définie les valeurs *nodata* au niveau de la bande vrt (différentes valeurs peuvent être fournies pour chaque bande). Si plus d'une valeur est fournie toutes les valeurs doivent être entre guillemet pour les réunir ensemble comme argument unique. Si l'option n'est pas définie, les définitions *nodata* intrinsèques sur le premier jeu de donnée seron utilisé (s'ils existent). La valeur définie par cette option est écrit dans l'élément *NoDataValue* de chaque élément *VRTRasterBand*. Utilisez une valeur à *None* pour ignorer les définitions *nodata* intrinsèques des jeux de données sources.
- **-separate :** (à partir de GDAL 1.7.0) place chaque fichier en entré dans une bande séparée et empilée. Dans ce cas, seule la première bande de chaque jeu de données sera placé dans une nouvelle bande. Contrairement au mode par défaut, il ne nécessite pas que toutes les bandes aient le même type de données.
- **-allow_projection_difference :** (à partir de GDAL 1.7.0) quand cette option est définie, la commande acceptera de réaliser un VRT même si les jeux de données en entrée n'ont pas la même projection. Note : cela ne signifie pas qu'ils seront

reprojeté. Leurs projections seront simplement ignoré.
- **-input_file_list :** pour définir un fichier texte avec un nom de fichier à chaque ligne.
- **-q :** (à partir de GDAL 1.7.0) pour désactiver la barre de progression dans la console.
- **-overwrite :** écrase le VRT s'il existe déjà.

3 Exemple

```
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
```

H gdal_contour

construit les lignes de contours à partir d'un modèle d'élévation raster dans un fichier vecteur.

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>
```

Ce programme génère un fichier de contour en vecteur à partir d'un modèle d'élévation en raster (DEM).

À partir de la version 1.7 les chaînes linéaire de contour seront orientées systématiquement. Le côté haut sera vers la droite, c'est à dire qu'une ligne aura pour sens le sens des aiguilles d'une montre.

- **-b band :** définie une bande particulière à utiliser à partir du DEM. Par défaut la bande nº 1.
- **-a name :** fournie un nom pour l'attribut dans lequel placer l'élévation. S'il n'est pas fourni, aucun attribut d'élévation n'est attaché.
- **-3d :** force la production de vecteurs 3D à la place e 2D. En incluant l'élévation à chaque vertex.
- **-inodata :** ignore les valeurs nodata implicites dans l'ensemble de données – traite toutes les valeurs comme valides.
- **-snodata value :** valeur des pixels en entrée à traiter en tant que « nodata ».
- **-f format :** créé une sortie dans un format particulier, en shapefile par défaut.
- **-i interval :** intervalle des élévations entre les contours.
- **-off offset :** début du zéro relatif à partir duquel interpréter les intervalles.
- **-fl level :** nomme un ou plusieurs « niveaux fixés » à extraire.
- **-nln outlayername :** fournie un nom pour la couche vectorielle en sortie. *contour* par défaut.

Exemple :

Cela créera un contour de 10 m à partir des données d'un DEM dans le fichier *dem.tif* et

produira un shapefile dans *contour.shp*, *contour.shx*, *contour.dbf* avec les élévations des contours dans l'attribut « elev ».

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

I gdaldem

Outils pour analyser et visualiser des MNT, (à partir de GDAL 1.7.0)

1 Synopsis

Usage :

- Pour générer une carte de reliefs ombragés à partir de n'importe quel raster d'élévation géré par GDAL :

```
gdaldem hillshade input_dem output_hillshade
        [-z ZFactor (défaut=1)] [-s scale* (défaut=1)]"
        [-az Azimuth (défaut=315)] [-alt Altitude (défaut=45)]
        [-alg ZevenbergenThorne]
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-co
"NAME=VALUE"]* [-q]
```

- Pour générer une carte des pentes à partir de n'importe quel raster d'élévation géré par GDAL :

```
gdaldem slope input_dem output_slope_map"
        [-p use percent slope (défaut=degrees)] [-s scale*
(défaut=1)]
        [-alg ZevenbergenThorne]
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-co
"NAME=VALUE"]* [-q]
```

- Pour générer une carte d'aspect à partir de n'importe quel raster d'élévation géré par GDAL. Créé en sortie un raster en float de 32 bit avec des valeurs de pixels de 0 à 360 indiquant l'azymuth :

```
gdaldem aspect input_dem output_aspect_map"
        [-trigonometric] [-zero_for_flat]
        [-alg ZevenbergenThorne]
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-co
"NAME=VALUE"]* [-q]
```

- Pour générer une carte de relief en couleur à partir de n'importe quel raster d'élévation géré par GDAL où *color_text_file* contient les lignes au format "elevation_value rouge vert bleu" :

```
gdaldem color-relief input_dem color_text_file
output_color_relief_map
        [-alpha] [-exact_color_entry | -nearest_color_entry]
        [-b Band (défaut=1)] [-of format] [-co "NAME=VALUE"]* [-
q]
```

- Pour générer une carte d'Index de Rugosité du Terrain (*Terrain Ruggedness Index* (TRI)) à partir de n'importe quel raster d'élévation géré par GDAL :

```
gdaldem TRI input_dem output_TRI_map
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-q]
```

- Pour générer une carte d'Index de Position Topographique (*Topographic Position Index* (TPI)) à partir de n'importe quel raster d'élévation géré par GDAL :

```
gdaldem TPI input_dem output_TPI_map
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-q]
```

- Pour générer une carte de rugosité à partir de n'importe quel raster d'élévation géré par GDAL :

```
gdaldem roughness input_dem output_roughness_map
        [-compute_edges] [-b Band (défaut=1)] [-of format] [-q]
```

Note!

Scale est le rapport des unités verticales sur celles horizontales pour Feet:Latlong utilisez scale=370400, pour Meters:LatLong utilisez scale=111120

Cette commande possède 7 modes différents :

- **hillshade :** pour générer une carte de relief ombragé à partir de n'importe quel raster d'élévation géré par GDAL.
- **slope :** pour générer une carte des pentes à partir de n'importe quel raster d'élévation géré par GDAL.
- **aspect :** pour générer une carte d'aspect à partir de n'importe quel raster d'élévation géré par GDAL.
- **color-relief :** pour générer une carte des reliefs en couleur à partir de n'importe quel raster d'élévation géré par GDAL.
- **TRI :** pour générer une carte d'Index de Rugosité du Terrain à partir de n'importe quel raster d'élévation géré par GDAL.
- **TPI :** pour générer une carte d'Index de Position Topographique à partir de n'importe quel raster d'élévation géré par GDAL.
- **roughness :** pour générer une carte de rugosité à partir de n'importe quel raster d'élévation géré par GDAL.

Les options générales suivantes sont disponibles :

- **input_dem :** le raster MNT en entrée à traiter.
- **output_xxx_map :** le raster en sortie produit.
- **-of format :** sélectionne le format de sortie. La valeur par défaut est GeoTIFF (GTiff). Utiliser le nom court du format.
- **-compute_edges :** (GDAL >= 1.8.0) réalise le calcul au bord du raster et proche des valeurs *nodata*
- **-alg ZevenbergenThorne :** (GDAL >= 1.8.0) utilise les formules de Zevenbergen & Thorne au lieu de celle de Horn pour calculer les pentes et les aspects. La littérature suggère que celle de Zevenbergen & Thorne est plus adaptée aux paysages doux, tandis que celle de Horn a de meilleurs résultats sur les terrains plus rudes.
- **-b band :** sélectionne une bande en entrée à traiter. Les bandes sont numérotées à partir de 1.

- **-co "NAME=VALUE" :** passe une option de création au pilote du format de sortie. Plusieurs options *-co* peuvent être listées. Voyez la documentation spécifique du format pour les options de création légales pour chaque format.
- **-q :** supprime la barre de progression et les autres informations.

Pour tous les algorithme, sauf *color-relief*, une valeur *nodata* dans le jeu de données cible sera émise si au moins un pixel définie dans la valeur *nodata* est trouvé dans une fenêtre 3x3 centrée autour du pixel source. La conséquence de cela est qu'il y aura un bord d'un pixel autour de chaque image définie à la valeur *nodata*. À partir de GDAL 1.8.0, si l'option *-compute_edges* est définie, ''gdaldem'' calculera les valeurs au bord de l'image ou si une valeur *nodata* est trouvée dans la fenêtre 3x3, en interpolant les valeurs manquantes.

2 Modes

2.a hillshade

Cette commande renvoie un raster de 8 bit avec un bel effet de relief ombragé. Cela est très utile pour visualiser le terrain. Vous pouvez en option définir l'azymuth et l'altitude la source de lumière, un facteur d'exagération vertical et un facteur d'échelle à tenir compte pour les différences entre les unités verticales et horizontales.

La valeur 0 est utilisée comme valeur *nodata* en sortie.

Les options spécifiques suivantes sont disponibles :

- **-z zFactor :** exagération verticale utilisé pour multiplier les élévations.
- **-s scale :** rapport des unités verticales et horizontale. Si l'unité horizontale du MNT source est le degrés (par exemple la projection Lat/Long WGS84), vous pouvez utiliser *scale=111120* si l'unité vertical sont les mètres (ou *scale=370400* si elles sont en pied).
- **-az azimuth :** azymuth de la lumière, en degrés. 0 si elle arrive d'en haut du raster, 90 de l'est, ... La valeur par défaut, 315, devrait rarement être changée puisque c'est la valeur généralement utilisé pour générer des cartes de relief.
- **-alt altitude :** altitude de la lumière, en degrés. 90 si la lumière arrive au dessus du MNT. 0 si c'est une lumière rasante.

2.b slope

Cette commande utilise un raster MNT et renvoie un raster en float 32 bit avec des valeurs de pente. Vous avez la possibilité de définir le type de pente que vous voulez : degrés ou pourcentage. Dans le cas où les unités horizontales diffèrent des unités verticales vous pouvez également fournie un facteur d'échelle.

La valeur -9999 est utilisé comme valeur *nodata* en sortie.

Les options spécifiques suivantes sont disponibles :

- **-p :** si définie, la pente sera exprimée en pourcent. Autrement, elle sera exprimée en degrés.
- **-s scale :** rapport des unités verticale et horizontale. Si l'unité horizontal du MNT source est le degrés (par exemple une projection WGS84 Lat/Long), vous pouvez utiliser *scale=111120* si l'unité vertical est le mètre (ou *scale=370400* si elles sont en pied).

2.c aspect

Cette commande renvoie un raster float 32 bit avec des valeurs entre 0° et 360° représentant l'azymuth dont les pentes font face. La définition de l'azymuth est : 0° signifie que la pente est face au Nord, 90° face à l'Est, 180° face au sud et 270° face à l'Ouest (en supposant que le haut du raster en entrée est orienté au Nord). La valeur -9999 de l'aspect est utilisé comme valeur *nodata* pour indiquer un aspect indéfinie dans les zones plates avec la pente = 0.

Les options spécifiques suivantes sont disponibles :

- **-trigonometric :** renvoie un angle trigonométrique au lieu de l'azymuth. Donc 0° signifie l'Est, 90° le Nord, 180° l'Ouest et 270° le Sud.
- **-zero_for_flat :** renvoie 0 pour les zones plates avec *slope=0* au lieu de -9999.

En utilisant ces deux options, l'aspect renvoyé par ''gdaldem aspect'' doit être identique à celui de la commande ''r.slope.aspect'' de GRASS. Autrement il est identique à celui de la commande *aspect.cpp* de Matthew Perry.

2.d color-relief

Cette commande renvoie un raster à 3 bande (RVB) ou à 4 bandes (RVBA) avec des valeurs calculées à partir de l'élévation et d'un fichier de configuration de couleur au format texte, contenant l'association entre les différentes valeurs d'élévation et la couleur désirée correspondante. Par défaut, les couleurs entre les valeurs d'élévation données sont mélangées en douceur et le résultat est un beau MNT coloré. Les options *-exact_color_entry* ou *-nearest_color_entry* peuvent être utilisées pour éviter cette interpolation linéaire pour les valeurs qui n'ont pas de correspondance avec un index de couleur du fichier de configuration.

Les options spécifiques suivantes sont disponibles :

- **color_text_file :** fichier de configuration des couleurs au format texte.
- **-alpha :** ajoute un canal alpha au raster en sortie.
- **-exact_color_entry :** utilise une stricte correspondance lors de la recherche dans le fichier de configuration des couleurs. Si aucun couleur correspondante n'est trouvée, le quadruplet RVBA "0,0,0,0"sera utilisé.
- **-nearest_color_entry :** utilise le quadruplet RVBA correspondant à l'entrée le plus proche dans le fichier de configuration des couleurs.

Le mode *color-relief* est le seul mode gérant le format VRT en sortie. Dans ce cas, il traduira le fichier de configuration des couleurs en éléments <LUT> appropriés. Notez que les élévations définie en pourcentage seront traduit en valeur absolue, ce qui doit être pris en compte lorsque les statistiques du raster source diffère de celui qui a été utilisé lors de la construction du VRT.

Le fichier de configuration des couleurs au format texte contient généralement 4 colonnes par ligne : la valeur de l'élévation et les composants correspondants de Rouge, Vert, Bleu (entre 0 et 255). La valeur de l'élévation peut être une valeur en virgule flottante, ou le mot-clé *nv* pour la valeur *nodata*. L'élévation peut aussi être exprimée en pourcentage : 0 % étant la valeur minimale trouvé dans le raster, 100 % la valeur maximale.

Une colonne supplémentaire peut être ajouté optionnellement pour le composant alpha. S'il n'est pas définie, l'opacité complète (255) est supposée.

Différents séparateurs de champs sont acceptés : virgule, tabulation, espaces, ':'.

Les couleurs communes utilisées par GRASS peuvent également être spécifiées en

utilisant leur nom, au lieu du triplet RVB. La liste des noms gérés est : *white*, *black*, *red*, *green*, *blue*, *yellow*, *magenta*, *cyan*, *aqua*, *grey/gray*, *orange*, *brown*, *purple/violet* et *indigo*.

Depuis GDAL 1.8.0, les fichiers de palette .cpt GMT sont également géré (COLOR_MODEL = RGB suelement).

Note!

La syntaxe du fichier de configuration de couleur est dérivé de celui géré par la commande r.colors de GRASS. Les fichiers (.clr) de table de couleur HDR d'ESRI correspondent également à cette syntaxe. Le composent alpha et la gestion des tabulations et virgules comme séparateurs sont des extensions spécifiques à GDAL.

Par exemple :

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

2.e TRI

Cette commande renvoie un raster à une seule bande avec des valeurs calculées à partir de l'élévation. TRI signifie *Terrain Ruggedness Index*, qui est définie comme la différence moyenne entre un pixel central et ses cellules l'entourant (voir *Wilson et al 2007, Marine Geodesy 30:3-35*).

La valeur -9999 est utilisé comme valeur *nodata* en sortie.

Il n'y a pas d'options spécifiques.

2.f TPI

Cette commande renvoie un raste à une seule bande avec des valeurs calculées à partir de l'élévation. TPI signifie *Topographic Position Index*, qui est définie comme la différence entre un pixel central et la moyenne des cellules l'entourant (voir *Wilson et al 2007, Marine Geodesy 30:3-35*).

La valeur -9999 est utilisé comme valeur *nodata* en sortie.

Il n'y a pas d'options spécifiques.

2.g roughness

Cette commande renvoie un raster à une seule bande calculé à partir de l'élévation. La rugosité est la plus grande différence inter-cellule d'un pixel central et ses cellules l'entourant, comme définie dans *Wilson et al (2007, Marine Geodesy 30:3-35)*.

La valeur -9999 est utilisé comme valeur *nodata* en sortie.

Il n'y a pas d'options spécifiques.

3 Auteurs

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.

4 Voir également

Documentation des commandes GRASS connexes :

- 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

J rgb2pct.py

Convertit une image RVB (24 bits) en une image en pseudo-couleurs (8 bits).

Usage :

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

Cette commande calculera une table pseudo-couleur optimale pour une image RVB donnée en utilisant un algorithme de coupe médian à partir d'un histogramme RVB ré-échantillonné. Puis il convertit l'image en image pseudo-couleur en utilisant la table de couleur. Cette conversion utilise l'erreur de diffusion Floyd-Steinberg pour optimiser la qualité visuelle de l'image en sortie.

- **-n colors :** sélectionne le nombre de couleurs dans la table de couleurs générée. Par défaut il est défini à 256. Il doit être compris entre 2 et 256.
- **-pct palette_file :** extrait la table de couleur à partir de *palette_file* au lieu de la calculer. Elle peut être utilisée pour obtenir une table de couleur cohérente pour des fichiers multiples. *palette_file* doit 3 être un fichier raster dans un format géré par GDAL avec une palette.
- **-of format :** format à générer (par défaut un GéoTIFF). Même sémantique que l'option *-of* pour ''gdal_translate''. Seuls les formats de sortie supportant les tables de pseudo-couleurs doivent être utilisés.
- **source_file :** le fichier RVB en entrée.
- **dest_file :** le fichier en pseudo-couleur en sortie qui sera créé.

Note!

REMARQUE : *''rbg2pct.py'' est un script Python, et fonctionnera seulement si GDAL a été compilé avec le support de Python.*

1 Exemple

Si vous désirez créer la palette à la main, le format le plus simple est probablement le format VRT de GDAL. Dans l'exemple suivant un VRT a été créé dans un éditeur de texte avec une petite palette de 4 couleurs avec les couleurs RVB 238/238/238/255, 237/237/237/255, 236/236/236/255 et 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>
```

K pct2rgb.py

Convertit une image en pseudo-couleurs (8 bits) en une image RVB (24 bits)

Usage :

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

Cette commande convertira une bande en pseudo-couleur d'un fichier en entrée en un fichier RVB en sortie au format désiré.

- **-of format :** format à générer (par défaut un GeoTIFF).
- **-b band :** bande à convertir en RVB, 1 par défaut.
- **-rgba :** générer un fichier RVBA (au lieu d'un fichier RVB par défaut).
- **source_file :** le fichier en entrée.
- **dest_file :** le fichier en RVB en sortie qui sera créé.

Note!

REMARQUE : *''rbg2pct.py'' est un script Python, et fonctionnera seulement si GDAL a été compilé avec le support de Python.*

Warning!

La nouvelle option '-expand rgb|rgba' de l'utilitaire ''gdal_translate'' rend ce script obsolète.

L gdal_merge.py

Mosaïquage d'un ensemble d'images

Usage :

```
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
```

Cette commande mosaïquera automatiquement un ensemble d'images. Toutes les images devront être dans le même système de coordonnées et avoir un nombre assorti de bandes, mais elles peuvent se superposer et avoir des résolutions différentes. Dans les zones de superposition, la dernière image sera copiée par dessus les premières.

- **-o out_filename :** nom du fichier en sortie, qui sera créé s'il n'existe pas (*out.tif* par défaut).
- **-of format :** format de sortie, GéoTIFF par défaut (GTiff).
- **-co NAME=VALUE :** option de création pour le fichier de sortie. Des options multiples peuvent être définies.
- **-ot datatype :** force un type défini pour la sortie des bandes d'images. Utilisez les noms des types (c'est-à-dire Byte, Int16...).
- **-ps pixelsize_x pixelsize_y :** taille du pixel à utiliser pour le fichier en sortie. S'il n'est pas défini, la résolution du premier fichier sera utilisée.
- **-tap :** (GDAL >= 1.8.0) (*target aligned pixels*) aligne les coordonnées de l'étendue du fichier en sortie aux valeurs de l'option *-tr* de telle sorte que l'étendue alignée inclue l'étendue minimale.
- **-ul_lr ulx uly lrx lry :** l'étendu du fichier de sortie. S'il n'est pas défini, la somme de toutes les étendues sera utilisée.
- **-v :** génère une sortie bavarde des opérations de mosaïcage lorsqu'elles sont réalisées.
- **-separate :** chaque fichier en entrée dans une bande empilée séparée.
- **-pct :** récupérer la table pseudo-couleur à partir du premier fichier image en entrée, et l'utiliser pour la sortie. Fusionner les images en pseudo-couleurs de cette façon assume que tous les fichiers en entrée ont la même table de couleur.
- **-n nodata_value :** ignore les pixels des fichiers qui sont fusionnés égaux à cette valeur du pixel.
- **-init value(s) :** pré-initialise la bande du fichier en sortie avec ces valeurs. Cependant, elle n'est pas notée comme valeur *nodata* dans le fichier en sortie. Si une seule est données, la même valeur sera utilisé dans toutes les bandes.
- **-createonly :** le fichier en sortie est créé (et éventuellement pré-initialisé) mais aucune image en entrée n'est copié dans celui-ci.

Warning!

rbg2pct.py est un script Python, et ne fonctionnera seulement si GDAL a été compilé avec le support de Python.

1 Exemple

Créer une image avec les pixels dans toutes les bandes initialisés à 255.

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

Créer une image RVB qui affiche en bleu les pixels sans données. Les deux premières bandes seront initialisé à 0 et la troisième le sera à 255.

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

M gdal2tiles.py

génère un répertoire avec des tuiles TMS, un KML et des visualisateurs web simples.

1 Usage

```
gdal2tiles.py [-title "Title"] [-publishurl http:*yourserver/dir/]
            [-nogooglemaps] [-noopenlayers] [-nokml]
            [-googlemapskey KEY] [-forcekml] [-v]
            input_file [output_dir]
```

2 Description

La commande génère un répertoire avec de petites tuiles et métadonnées, suivant la spécification du Service de Tuilage de carte de l'OSGeo. Des pages web simples avec des visualiseurs basés sur Google Map et OpenLayers sont générés également - tout le monde peut confortablement explorer vos cartes en ligne et vous n'avez pas besoin d'installer ou de configurer un logiciel spécial (comme Mapserver) et la carte s'affiche très rapidement dans le navigateur web. Vous avez seulement besoin de télécharger votre répertoire généré dans un serveur web.

`GDAL2Tiles` créé également les métadonnées nécessaires pour Google Earth (*SuperOverlay* KML), dans le cas où la carte fournie utilise une projection *EPSG:4326*.

Les fichiers world et les références spatiales incluses sont utilisés durant la génération des tuiles, mais vous pouvez publier également une image sans le géoréférencement.

- **-title "Title" :** titre utilisé pour les métadonnées générées, les visualisateurs web et les fichiers KML.
- **-publishurl http:*yourserver/dir/ :** adresse du répertoire dans lequel vous allez télécharger le résultat. Il doit se terminer par un slash.
- **-nogooglemaps :** ne génère pas de page HTML de base pour Google Maps.
- **-noopenlayers :** ne génère pas de page HTML de base pour OpenLayers.
- **-nokml :** ne génère pas de fichier KML pour Google Earth.
- **-googlemapskey KEY :** clé pour votre domaine généré sur la page web de l'API de Google Maps (http://www.google.com/apis/maps/signup.html).
- **-forcekml :** force la régénération des fichiers KML. Le fichier en entrée doit utiliser des coordonnées *EPSG:4326* !
- **-v :** génère une sortie verbeuse lors de la génération des tuiles.

Warning!

gdal2tiles.py est un script Python qui nécessite la compilation avec la liaison python de nouvelle génération.

N gdal_rasterize

Rastérise un vecteur polygone dans un raster.

Usage :

```
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>
```

Ce programme transforme des géométries vectorielles (points, lignes et polygones) dans des bande(s) raster d'une image raster. Les fichiers vecteurs sont lus à partir des formats vectoriels gérés par OGR.

Notez que les données vecteur doivent être dans le même système de coordonnées que les données raster ; la reprojection à la volée n'est pas possible.

Depuis GDAL 1.8.0, le fichier cible de GDAL peut être créé par `gdal_rasterize`. Une des options *-tr* ou *-ts* doit être utilisé dans ce cas.

- **-b band :** la bande dans laquelle placer les valeurs. Plusieurs arguments *-b* peuvent être utilisés pour transformer une liste de bandes. Par défaut, une seule bande est transformée.
- **-i :** inverse la rastérisation. Imposer la valeur de la brulure fixée ou la valeur de la brulure associée avec le premier objet dans toute l'image en dehors du polygone fournie.
- **-at :** active l'option de rasterisation ALL_TOUCHED afin que tous les pixels touchés par des lignes ou des polygones seront mis à jour et pas seulement ceux sur le chemin de la ligne ou dont le point centrale est dans le polygone. les règles de rendu normal sont désactivé par défaut.
- **-burn value :** une valeur fixe à créer dans la bande pour tous les objets. Une liste d'options -burn peut être fournit, un par bande à écrire.
- **-a attribute_name :** définit un champ d'attribut à utiliser sur l'objet comme valeur finale. Cette valeur sera utilisée dans toutes les bandes en sortie.
- **-3d :** indique que la valeur finale doit être extraite à partir de la valeur « Z » de l'objet (pas encore implémenté). Ces valeurs sont ajustées par la valeur de rasterisation donnée par les options "-burn value" ou "-a attribute_name" si ceux-ci ont été fournie. Pour l'instant seuls les points et les lignes sont dessinés en 3D.
- **-l layername :** la ou les couche(s) de la source de données qui sera utilisée pour les objets en entrées. Peut être définie plusieurs fois, mais au moins une couche ou une option -sql doit être définie.
- **-where expression :** une requête SQL de style WHERE optionnel doit être appliqué pour sélectionner les objets à rastériser à partir d'une ou plusieurs couche(s).
- **-sql select_statement :** requête SQL à utilisée sur la source de données pour produire une couche virtuelle d'objets à rastériser.
- **-of format :** (GDAL >= 1.8.0) sélectionne le format en sortie. GeoTIFF (GTiff) par défaut. Utilisez le nom de format court.
- **-a_nodata value :** (GDAL >= 1.8.0) assigne une valeur *nodata* définie aux bandes en sortie.
- **-init value :** (GDAL >= 1.8.0) Pré-initialise les bandes d'image en sortie avec ces valeurs. Cependant, elles ne sont pas noté comme valeur *nodata* dans le fichier en sortie. Si seulement une valeur est données, la même valeur est utilisée pour toutes les bandes.

- **-a_srs srs_def :** (GDAL >= 1.8.0) écrase la projection des fichiers en sortie. Si non spécifié, la projection du fichier vecteur en entrée sera utilisée si elle est disponible. Si les projections sont incompatible entre les fichiers en entrée et en sorties, aucune tentative de reprojection ne sera effectuée. La valeur de *srs_def* peut être n'importe quelle valeur sous la forme gérée par GDAL/OGR, WKT complet, PROJ.4, EPSG:n ou un fichier contenant le WKT.
- **-co "NAME=VALUE" :** (GDAL >= 1.8.0) passe une option de création au pilote du format de sortie. De multiples options *-co* peuvent être listées. Voyez la documentation spécifique du format pour les options de création légales pour chaque format.
- **-te xmin ymin xmax ymax :** (GDAL >= 1.8.0) définie les étendues géoréférencées. Les valeurs doivent être exprimées en unité du géoréférencement. Si aucun n'est définie, l'étendue du fichier en sortie sera l'étendue des couches vecteurs.
- **-tr xres yres :** (GDAL >= 1.8.0) définie la résolution cible. Les valeurs doivent être exprimées en unité de géoréférencement. Les deux valeurs doivent être positives.
- **-tap :** (GDAL >= 1.8.0) (*target aligned pixels*) aligne les coordonnées de l'étendue du fichier en sortie aux valeurs de l'option *-tr*, tel que l'étendue aligné inclue l'étendue minimale.
- **-ts width height :** (GDAL >= 1.8.0) définie la taille du fichier en sortie en pixels et lignes. Notez que l'option *-ts* ne peut pas être utilisé avec l'option *-tr*.
- **-ot type :** (GDAL >= 1.8.0) pour que les bandes en sortie soient du type de données indiqué. Float64 par défaut.
- **-q :** (GDAL >= 1.8.0) supprime la barre de progression et autre affichage d'information.
- **src_datasource :** n'importe quelle source de données supportée par OGR en lecture.
- **dst_filename :** le fichier de sortie supporté par GDAL. Doit supporter le mode d'accès de mise à jour. Avant la version 1.8.0 de GDAL, `gdal_rasterize` ne pouvait pas créer de nouveau fichier de sortie.

Exemples :

La commande suivante rastérisera tous les polygones à partir de mask.shp en un fichier RGB TIFF work.tif avec la couleur rouge (RGB = 255,0,0) :

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

La commande suivante rastérisera tout les bâtiments « class A » dans le fichier d'élévation en sortie, en prenant l'élévation à partir de l'attribut ROOF_H :

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

O gdaltransform

Transforme des coordonnées

Usage :

```
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]]
```

1 Description

La commande ''gdaltransform'' reprojete une liste de coordonnées vers n'importe quelle projection gérée, incluant les transformations basées sur les points d'amer (GCP).

- **-s_srs srs def :** définition de la référence spatiale de la source. Le système de coordonnées qui peut être envoyé est n'importe lequel géré par l'appel *OGRSpatialReference.SetFromUserInput()*, ce qui inclut EPSG PCS et points d'amer (c'est-à-dire EPSG:4296), les déclarations PROJ.4 (comme ci-dessus), ou le nom d'un fichier .prj contenant un format Well Known Text.
- **-t_srs srs_def :** définition de la référence spatiale de la cible. Le système de coordonnées qui peut être envoyé est n'importe lequel géré par l'appel *OGRSpatialReference.SetFromUserInput()*, ce qui inclut EPSG PCS et points d'amer (c'est-à-dire EPSG:4296), les déclarations PROJ.4 (comme ci-dessus), ou le nom d'un fichier .prj contenant un format Well Known Text.
- **-to NAME=VALUE :** définie une option de transformation disponible à envoyer à *GDALCreateGenImgProjTransformer2()*.
- **-order n :** ordre de la fonction polynomiale utilisé pour le découpage (1 à 3). Par défaut la valeur est basée sur le nombre de points d'amer disponible.
- **-tps :** force l'utilisation d'une transformation par interpolation (thin plate spline) basée sur les points d'amer disponibles.
- **-rpc :** force l'utilisation des RPCs.
- **-geoloc :** force l'utilisation des tableaux de géolocation.
- **-i :** transformation inverse : de la destination vers la source.
- **-gcppixel line easting northing [elevation] :** fourni un point d'amer à utiliser pour la transformation (généralement ou plus sont requis).
- **srcfile :** fichier avec la définition de la projection source ou les points d'amer. S'il n'est pas donné, la projection source est lu à partir de la ligne de commande *-s_srs* ou les paramètres *-gcp*.
- **dstfile :** fichier avec la définition de la projection de destination.

Les coordonnées sont lues par pair (ou triplet) de nombre par ligne de l'entrée standard, transformée, et écrite vers la sortie standard de la même manière. Toutes les transformations offertes par ''gdalwarp'' sont prises en charge, incluant celles basées sur les points d'amer.

Notez que l'entrée et la sortie doivent toujours sous forme décimale. Il n'y pas de gestion actuellement pour l'entrée et la sortie en DMS.

Si un fichier image en entré est fournie, input est en coordonnées pixel/ligne sur cette image. Si un fichier sorti est fourni, l'output est en coordonnées pixel/ligne sur cette image.

2 Exemple de reprojection

Simple reprojection d'un système de coordonnées projeté à un autre :

```
gdaltransform -s_srs EPSG:28992 -t_srs EPSG:31370
177502 311865
```

Produit la sortie suivante en mètre dans la projection "Belge 1972 / Belgian Lambert 72" :

```
244510.77404604 166154.532871342 -1046.79270555763
```

3 Exemple d'image RPC

La commande suivante demande une transformation basée sur des RPC en utilisant le modèle RPC associé avec le fichier nommé. Parce que l'option *-i* (inverse) est utilisée, la transformation part des coordonnées géoréférencées (WGS84) en sortie vers des coordonnées d'images.

```
gdaltransform -i -rpc 06OCT20025052-P2AS-005553965230_01_P001.TIF
125.67206 39.85307 50
```

produit cette sortie mesurée en pixel et lignes d'une image :

```
3499.49282422381 2910.83892848414 50
```

P nearblack

Convertie des bords plus ou moins en noirs/blanc en noir.

1 Usage

```
nearblack [-of format] [-white] [-near dist] [-nb non_black_pixels]
      [-setalpha] [-setmask] [-o outfile] [-q]  [-co "NAME=VALUE"]*
infile
```

2 Description

La commande va scanner une image et tenter de définir tous les pixels qui sont proches du noir (ou proche du blanc) autour du bord en noir (ou blanc) exact. Cela est souvent utilisé pour corriger les photos aériennes compressées avec perte afin que les pixels de couleur puissent être traités comme transparents lors du mosaïquage.

- **-o outfile :** le nom du fichier en sortie à créer. Les fichiers nouvellement créés sont toujours créés avec le pilote HFA (Erdas Imagine - .img) par défaut.
- **-of format :** (GDAL 1.8.0 ou supérieur) sélectionne le format en sortie. Utilisez le nom de format court (GTiff pour GeoTIFF par exemple).
- **-co "NAME=VALUE" :** (GDAL 1.8.0 ou supérieur) passe une option de création au pilote du format en sortie. Plusieurs options *-co* peuvent être listées. Voyez la documentation spécifique au format pour les options de création légales pour chaque format. Seulement valide lors de la création d'un nouveau fichier.

- **-white :** recherche pour les pixels proche du blanc (255) au lieu des pixels proche du noir.
- **-near dist :** sélectionne la distance du noir (ou du blanc) les valeurs du pixel peuvent être encore considéré comme noir (ou blanc). 15 par défaut.
- ** -nb non_black_pixels :** nombre de pixels différent du noir qui peut être rencontré avant d'abandonner la recherche à l'intérieur. 2 par défaut.
- **-setalpha :** (GDAL 1.8.0 ou supérieur) ajoute une bande alpha si le fichier en sortie est définie et si le fichier en entrée possède trois bandes, ou définie la bande alpha du fichier en sortie s'il est définie et le fichier en entrée possède quatre bandes, ou bien définie la bande alpha du fichier en entrée s'il possède 4 bandes et aucun fichier en sortie n'est définie. La bande alpha est définie à 0 dans le collier de l'image et à 255 partout ailleurs.
- **-setmask :** (GDAL 1.8.0 ou supérieur) ajoute une bande de masque au fichier en sortie ou ajoute une bande de masque au fichier en entrée s'il n'en a pas un déjà et aucun fichier n'est définie. La bande de masque est définie à 0 dans le collier de l'image et à 255 partout ailleurs.
- **-q :** (GDAL 1.8.0 ou supérieur) supprime la barre de progression et les autres informations affichées.
- **infile :** le fichier en entrée. N'importe quel format géré par GDAL, de n'importe quel nombre de bandes, normalement des bandes sous 8 bytes.

L'algorithme traite l'image une ligne à la foi. Un scan intérieur est réalisé à partir du pixel défini à la fin à noir (blanc) jusqu'à au moins "*non_black_pixels*" pixels qui sont éloignés d'au moins *dist* niveaux de noir (blanc) rencontré avant l'arrêt du scan. Les pixels proches du noir (blanc) sont définis à noir(blanc). L'algorithme scan également du haut vers le bas puis du bas vers le haut pour identifier les indentations dans le haut ou le bas.

Le traitement est entièrement réalisé en 8 bits (Bytes).

Si le fichier en sortie est omis, le résultat calculé sera écrit dans le fichier en entrée - qui doit gérer la mise à jour.

Chapitre III gdal_retile.py

`gdal_retile` - `gdal_retile.py` recréé un ensemble de tuiles et/ou construit les niveaux de la pyramide tuilée.

A Usage

Usage :

```
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
```

B Description

La commande recréera les tuiles d'un ensemble de tuiles en entrée. Toutes les tuiles en entrée doivent être géoréférencées dans le même système de coordonnées et avoir un nombre de bandes correspondant. En option les niveaux de pyramide sont générés. Il est possible de générer des fichiers shape pour la sortie tuilés.

Si votre nombre de fichiers d'entrée dépasse le buffer de la ligne de commande, utilise l'option générale `--optfile`.

- **-targetDir directory :** le répertoire dans lequel les tuiles résultantes seront créées. Les pyramides sont rangées dans des sous-répertoires numérotés à partir de 1. Les tuiles créées ont un schéma de numérotation et contienne le nom de(s) tuile(s) source ;
- **-of format :** format de sortie, GeoTIFF par défaut (GTiff).

- **-co NAME=VALUE :** option de création pour le fichier de sortie. Des options multiples peuvent être définies.
- **-ot datatype :** force les bandes d'images en sortie à un type spécifique. Utilisez le nom du type (c'est-à-dire Byte, Int16...)
- **-ps pixelsize_x pixelsize_y :** taille du pixel à utiliser pour le fichier de sortie. S'il n'est pas défini, un pixel de 256 x 256 est choisi par défaut ;
- **-levels numberOfLevels :** nombre de niveaux de pyramide à construire ;
- **-v :** génère une sortie verbeuse des opérations de tuilage lors de leur génération ;
- **-pyramidOnly :** pas de retuilage, construire seulement les pyramides ;
- **-r algorithm :** algorithme d'échantillonnage, *near* par défaut ;
- **-s_srs srs_def :** référence spatiale source à utiliser. Le système de coordonnées qui doit être envoyé peut avoir n'importe quel format géré par l'appel de *OGRSpatialReference.SetFro-mUserInput()*, ce qui inclut les EPSG PCS et GCSes (par exemple EPSG:4296),, les déclarations PROJ4 (comme ci-dessus), ou le nom d'un fichier .prj contenant une projection au format Well Known Text. Si aucun *srs_def* n'est donné, le *srs_def* des tuiles sources est utilisé (s'il en a un). Le *srs_def* sera propagé pour créer les tuiles (si possible) et au(x) fichier(s) optionnel(s) ;
- **-tileIndex tileIndexName :** le nom du fichier shape contenant l'index des tuiles résultantes ;
- **-tileIndexField tileIndexFieldName :** le nom de l'attribut contenant le nom des tuiles ;
- **-csv csvFileName :** le nom du fichier csv contenant les informations de géoréférencement des tuiles. Le fichier contient 5 colonnes : *tilename,minx,maxx,miny,maxy*
- **-csvDelim column delimiter :** le délimiteur de colonne utilisé dans le fichier csv, la valeur par défaut est le point-virgule ";".
- **-useDirForEachRow :** normalement les tuiles de l'image de base sont stockées comme décrites dans l'option *-targetDir*. Pour de grandes images, certains systèmes de fichier ont des problèmes de performances si le nombre de fichier dans un répertoire est trop important, ne permettant pas à `gdal_retile.py` de finir dans un temps raisonnable. Utiliser ce paramètre permet de créer une structure différente en sortie. Les tuiles de l'image de base sont stockées dans un sous-répertoire nommé 0, les pyramides dans les sous-répertoires sont numérotés 1,2,... Dans chacun des ces répertoires un autre niveau de sous-répertoire est créé, numéroté de 0 à n, en fonction du nombre de ligne de tuile qui sont nécessaire pour chaque niveau. Enfin, un répertoire contient seulement les tuiles pour une ligne d'un niveau spécifique. Pour les grandes images une amélioration de la performance d'un facteur N est réalisée.

Warning!

`gdal_merge.py` est un script Python, et ne fonctionnera seulement si GDAL a été compilé avec la gestion de Python.

Chapitre IV gdal_grid

créer une grille régulière à partir de données éparses.

A Usage

Usage :

```
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>
```

B Description

Ce programme créé une grille régulière (raster) à partir des données éparpillées à partir d'une source de données OGR. Les données en entrées seront interpolées pour remplir les nœuds de la grille avec des valeurs, vous pouvez choisir diverses méthodes d'interpolations.

- **-ot type :** pour définir les types des données des bandes en sortie.
- **-of format :** sélectionne le format de sortie. GeoTIFF par défaut (GTiff). Utiliser le nom du format court.
- **xmin xmax :** définie les étendues X géoréférencées du fichier en sortie à créer.
- **-tye ymin ymax :** définie les étendues Y géoréférencées du fichier en sortie à créer.
- **-outsize xsize ysize :** définie la taille du fichier en sortie en pixel et lignes.
- **-a_srs srs_def :** écrase la projection pour le fichier en sortie. Le *srs_def* peut être de n'importe quel format que ceux habituelle dans GDAL/OGR, WKT complet,

PROJ.4, EPSG:n ou un fichier contenant une projection au format WKT.

- **-zfield field_name :** identifie un champ attributaire sur l'objet à utiliser pour obtenir la valeur du z. Cette valeur écrase la valeur du Z lu dans la géométrie (naturellement, si vous avez une valeur Z dans la géométrie, sinon vous n'avez pas le choix et devez définir un nom de champ contenant la valeur du Z).
- **-a [algorithm[:parameter1=value1][:parameter2=value2]...] :** définie l'algorithme d'interpolation ou le nom de la métrique de données et (en option) leurs paramètres. Lisez la section :ref:`gdal.gdal.gdal_grid.algo` et :ref:`gdal.gdal.gdal_grid.metriques` pour plus d'information sur les options disponibles.
  Unknown interpreted text role "ref".
  Unknown interpreted text role "ref".
- **-spat xmin ymin xmax ymax :** ajoute un filtre spatial pour sélectionner seulement les géométries intersectant la boîte englobante décrite par (xmin, ymin) - (xmax, ymax).
- **-clipsrc [xmin ymin xmax ymax]|WKT|datasource|spat_extent :** ajoute un filtre spatial pour sélectionner seulement les features contenus dans une bouding box définie (exprimé dans le SRS source), une géométrie WKT (POLYGON ou MULTIPOLYGON), à partir d'une source de données ou bien de l'étendue spatiale de l'option *-spat* si vous utilisez le mot-clé *spat_extent*. Lorsque vous spécifiez une source de données, vous voudrez généralement l'utiliser en combinaison de l'option *-clipsrclayer*, *-clipsrcwhere* ou *-clipsrcsql*.
- **-clipsrcsql sql_statement :** sélectionne les géométries désirées en utilisant une requête SQL à la place.
- **-clipsrclayer layername :** sélectionne la couche nommée à partir de la source de données pour la découpe.
- **-clipsrcwhere expression :** restreint les géométries désirées basé sur une requête attributaire.
- **-l layername :** indique la couche de la source de données qui sera utilisée pour les objets en entrée. peut être définie de multiples fois, mais au moins une couche ou une option *-sql* doit être définie.
- **-where expression :** une expression SQL optionnelle de requête de style WHERE à appliquer aux objets sélectionnés à traiter à partir de la couche en entrée.
- **-sql select_statement :** une requête SQL à évaluer en fonction de la source de données pour produire une couche virtuelle d'objet à traiter.
- **-co "NAME=VALUE":** envoie une option de création au pilote de format de sortie. Des options *-co* multiple peuvent être listés. Lisez la documentation spécifique au format pour les options correctes de création pour chaque format.
- **-q :** supprime la visualisation de la progression et autres sorties qui ne sont pas des erreurs.
- **src_datasource :** n'importe quelle source de données lisibles gérées par OGR.
- **dst_filename :** le fichier de sortie géré par GDAL.

C Algorithmes d'interpolation

Il y a plusieurs algorithmes d'interpolation disponible.

1 invdist

Inverse de la distance à la puissance. C'est l'algorithme par défaut. Il possède les paramètres suivants :

- **power :** poids de la puissance (2.0 par défaut) ;

- **smoothing :** paramètre de douceur" (0.0 par défaut) ;
- **radius1 :** le premier rayon (axe des X si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **radius2 :** le second rayon (axe des Y si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **angle :** angle de rotation de l'ellipse de recherche en degré (sens des aiguilles d'une montre, 0.0 par défaut).
- **max_points :** nombre maximal de points de données à utiliser. Ne cherche pas un nombre de points au-dessus de ce nombre. Il est utilisé seulement si l'ellipse de recherche est définie (les deux rayons ne sont pas nul). Zéro signifie que tous les points trouvés doivent être utilisés. 0 est la valeur par défaut.
- **min_points :** nombre minimal de points de données à utiliser. Si un nombre de points inférieur est trouvé le noeud de la grille est considéré comme vide et sera rempli de valeur *nodata*. Il est utilisé seulement si la recherche de l'ellipse est définie (les deux rayons ne sont pas nuls). 0 est la valeur par défaut.
- **nodata :** valeur *NODATA* pour remplir les points vides (0.0 par défaut).

2 average

Algorithme de la moyenne mobile. Il possède les paramètres suivants :

- **radius1 :** le premier rayon (axe des X si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **radius2 :** le second rayon (axe des Y si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **angle :** angle de rotation de l'ellipse de recherche en degré (sens des aiguilles d'une montre, 0.0 par défaut).
- **min_points :** nombre minimal de points de données à utiliser. Si un nombre de points inférieur est trouvé le noeud de la grille est considéré comme vide et sera rempli de valeur *nodata*. 0 est la valeur par défaut.
- **nodata :** valeur *NODATA* pour remplir les points vides (0.0 par défaut).

Notez qu'il est essentiel de définir l'ellipse de recherche pour la méthode de la moyenne mobile. C'est une fenêtre qui sera moyennée lors du calcul des valeurs des nœuds de la grille.

3 nearest

Algorithme du plus proche voisin. Il possède les paramètres suivants :

- **radius1 :** le premier rayon (axe des X si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **radius2 :** le second rayon (axe des Y si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **angle :** angle de rotation de l'ellipse de recherche en degré (sens des aiguilles d'une montre, 0.0 par défaut).
- **nodata :** valeur *NODATA* pour remplir les points vides (0.0 par défaut).

D Métriques des données

Outre les fonctionnalités d'interpolation `gdal_grid can` peut être utilisé pour calculer certaines données métriques en utilisant la fenêtre définie et la géométrie grille en sortie. Ces métriques sont :

- **minimum :** valeur minimale trouvée dans l'ellipse de recherche du nœud de la grille.
- **maximum :** valeur maximale trouvée dans l'ellipse de recherche du nœud de la grille.
- **range :** une différence entre les valeurs minimales et maximales trouvées dans l'ellipse de recherche du nœud de la grille.
- **count :** un nombre de point trouvé dans l'ellipse de recherche de noeud de la grille.
- **average_distance :** une distance moyenne entre les noeuds de la grille (centre de l'ellipse de recherche) et toutes les données ponctuelles trouvé dans l'ellipse de recherche de noeud de la grille.
- **average_distance_pts :** une distance moyenne entre les données ponctuelles dans l'ellipse de recherche de noeud de la grille. La distance entre chaque pair de points dans l'ellipse est calculé et la moyenne de toutes les distances est définie comme valeur du noeud de la grille.

Tous les métriques ont les mêmes ensembles d'options :

- **radius1 :** le premier rayon (axe des X si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble . du tableau de point. 0.0 par défaut.
- **radius2 :** le second rayon (axe des Y si l'angle de rotation est 0) de l'ellipse de recherche. Définissez ce paramètre à 0 pour utiliser l'ensemble du tableau de point. 0.0 par défaut.
- **angle :** angle de rotation de l'ellipse de recherche en degré (sens des aiguilles d'une montre, 0.0 par défaut).
- **min_points :** nombre minimal de points de données à utiliser. Si un nombre de points inférieur est trouvé le noeud de la grille est considéré comme vide et sera rempli de valeur *nodata*. Il est utilisé seulement si la recherche de l'ellipse est définie (les deux rayons ne sont pas nuls). 0 est la valeur par défaut.
- **nodata :** valeur *NODATA* pour remplir les points vides (0.0 par défaut).

E Lire des valeurs séparées par des virgules

Souvent vous avez un fichier texte avec une liste de valeurs XYZ séparées par des virgules à utiliser (appelé fichier CSV). Vous pouvez facilement utiliser ce type de source de données dans *gdal_grid*. Tout ce que vous devez faire est de créer un en-tête de jeu de données virtuel (VRT) pour votre fichier CSV et l'utiliser comme jeu de données en entré pour *gdal_grid*. Vous pouvez trouver des détails supplémentaires sur la page de description du :ref:`gdal.ogr.formats.vrt`.

Unknown interpreted text role "ref".

Voici un petit exemple. Nous avons un fichier CSV appelé *dem.csv* contenant :

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

```
87077.6,891995,135.01
...
```

Pour les données ci-dessus nous créons un en-tête dem.vrt avec le contenu suivant :

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

Cette description définit une géométrie appelée 2.5D avec trois coordonnées X,Y et Z. La valeur Z sera utilisée pour l'interpolation. Maintenant vous pouvez utiliser le fichier *dem.vrt* avec tous les programmes OGR (démarrez avec `ogrinfo` pour tester que tout fonctionne correctement). La source de données contiendra une seule couche appelée "dem" rempli d'objet point construit à partir des valeurs contenu dans le fichier CSV. En utilisant cette technique, vous pouvez prendre en charge les fichiers CSV avec plus de trois colonnes, inverser des colonnes, etc.

Si votre fichier CSV ne contient pas d'en-tête de colonne alors il peut être pris en charge comme suit :

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

La page de description des fichiers :ref:`gdal.ogr.formats.csv` contient des détails sur le format CSV géré par GDAL/OGR.

Unknown interpreted text role "ref".

F Exemple

Les exemples suivants pourraient créer un fichier raster TIFF à partir d'une source de données VRT décrit dans la section :ref:`gdal.gdal.gdal_grid.csv` en utilisant la distance inverse à la puissance. Les valeurs à interpoler seront lues partir de la valeur Z de l'enregistrement de la géométrie.

Unknown interpreted text role "ref".

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

La commande suivante fait la même chose que la précédente, mais lit les valeurs pour interpoler à partir du champ attributaire défini avec l'option *-zfield* à la place de l'enregistrement des géométries. Dans ce cas donc, les coordonnées X et Y sont récupérées dans la géométrie et le Z du champ *Elevation*.

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

Chapitre V gdal_proximity

produit une carte raster de proximité.

A Usage

```
gdal_proximity.py srcfile dstfile [-srcband n] [-dstband n]
              [-of format] [-co name=value]*
              [-ot Byte/Int16/Int32/Float32/etc]
              [-values n,n,n] [-distunits PIXEL/GEO]
              [-maxdist n] [-nodata n] [-fixed-buf-val n]
```

B Description

Le script `gdal_proximty.py` génère une carter raster de proximité indiquant la distance du centre de chaque pixel au centre du pixel le plus proche identifié comme le pixel cible. Les pixels cibles sont ceux du raster source pour lequel la valeur du pixel du raster est dans l'ensemble des valeurs des pixels cibles.

- **srcfile :** le fichier raster source pour identifier les pixels cibles.
- **dstfile :** le fichier raster de destination dans laquelle la carte de proximité sera écrite.
- **-srcband n :** identifie la bande dans le fichier source à utiliser (1 par défaut).
- **-srcband n :** identifie la bande dans le fichier de destination à utiliser (1 par défaut).
- **-of format :** sélectionne le format de sortie. GeoTIFF par défaut (GTiff). Utilisez le nom du format court.
- **-co "NAME=VALUE" :** envoie une option de création au pilote du format de sortie. De multiplies options *-co* peuvent être listé. Lisez la documentation spécifique du format pour les options légales de création pour chaque format.
- **-ot datatype :** force le type spécifique de la bande d'images en sortie. Utilisez les noms de type (c'est-à-dire Byte, Int16...)
- **-values n,n,n :** une liste de valeurs de pixel cible dans l'image source à considérer comme des pixels cibles. Si elle n'est pas définie, tous les pixels différents de zéro seront considérés comme des pixels cibles.
- **-distunits PIXEL/GEO :** indique si les distances générées doivent être en pixel

ou en coordonnées géoéréférencées (PIXEL par défaut).

- **-maxdist n :** la distance maximale à générer. Tous les pixels au-delà de cette distance se verront assignés soit la valeur *nodata* soit 65535. La distance est interprétée en pixel à moins que le paramètre *-distunits GEO* ne soit définie.
- **-nodata n :** définie nue valeur *nodata* à utiliser pour le raster de proximité de destination.
- **-fixed-buf-val n :** définie une valeur à appliquer à tous les pixels qui sont à *-maxdist* des pixels cibles (pixels cibles inclut) à la place d'une valeur de distance.

Chapitre VI gdal_polygonize

Produit une couche polygonale à partir d'un raster.

Usage :

```
gdal_polygonize [-o name=value] [-nomask] [-mask filename] raster_file
[-b band]
            [-q] [-f ogr_format] out_file [layer] [fieldname]
```

Description :

Cette commande créée des polygones vectoriels pour toutes les zones connectés d'un pixel dans un raster partageant une valeur commune. Chaque polygone est créé avec un attribut indiquant la valeur du pixel de ce polygone. Un masque de raster peut aussi être fourni pour déterminer quels pixels sont éligibles pour le traitement.

La commande créera le jeu de données vectoriel en sortie si celui-ci n'existe pas, par défaut dans le format GML.

Elle est basée sur la fonction `GDALPolygonize()` qui possède des détails supplémentaires sur l'algorithme.

- **-nomask :** n'utilise pas le masque de validité par défaut pour la bande en entrée (tel que nodata, ou les masques alpha).
- **-mask filename :** utilise la première bande du fichier définie comme masque valide (zéro est invalide, non zéro est valide).
- **raster_file :** le fichier raster source à partir duquel les polygones sont dérivés.
- **-b band :** la bande de *raster_file* à partir de laquelle construire les polygones.
- **-f ogr_format :** sélectionne le format de sortie du fichier à créer. GML par défaut.
- **out_file :** le fichier vecteur de destination dans lequel les polygones seront écrits.
- **layer :** le nom de la couche à créer pour contenir les polygones.
- **fieldname :** le nom du champ à créer ("DN" par défaut).
- **-o name=value :** définie un argument spécial à l'algorithme. Pour l'instant aucun n'est géré.
- **-q :** le script fonctionne en mode silencieux. La barre de progression est supprimée et les messages du traitement ne sont pas affichés.

Chapitre VII gdal_sieve

Supprime les petits polygones raster.

Usage :

```
gdal_sieve [-q] [-st threshold] [-4] [-8] [-o name=value]
       srcfile [-nomask] [-mask filename] [-of format] [dstfile]
```

Description :

Le script `gdal_sieve.py` enlève les polygones raster plus petits que la taille du seuil définie (en pixels) et les remplace avec la valeur du pixel du polygone du plus proche voisin. Le résultat peut être écrit dans la bande raster existante ou copié dans un nouveau fichier.

Des détails supplémentaires sur l'algorithme sont disponibles dans la documentation `GDALSieveFilter()`.

- **-q :** le script se lance en mode silencieux. La barre de progression est supprimée et les messages ne sont pas affichés.
- **-st threshold :** définie le seuil de la taille en pixel. Seuls les polygones raster plus petits que cette taille seront supprimés.
- **-o name=value :** définie un argument spécial à l'algorithme. Pour l'instant non géré.
- **-4 :** 4 connectivités doivent être utilisées lors de la détermination des polygones. C'est-à-dire que les diagonales des pixels ne sont pas considérées comme connectées. C'est la valeur par défaut
- **-8 :** 8 connectivités doivent être utilisées lors de la détermination des polygones. C'est-à-dire que les diagonales des pixels sont considérées comme directement connectées.
- **srcfile :** le fichier raster source utilisé pour identifié les pixels cibles. Seule la première bande est utilisée.
- **-nomask :** n'utilise pas le masque de validité par défaut pour la bande en entrée (tel que nodata ou les masques alpha).
- **-mask filename :** utilise la première bande du fichier définie comme masque de validité (zéro est invalide, autre que zéro est valide).
- **dstfile :** le nouveau fichier à créer avec les résultats filtrés. S'il n'est pas fourni,

la bande source sera mise à jour à la place.

- **-of format :** sélectionne le format de sortie. GeoTIFF par défaut (GTiff). Utilisez le nom de format court.

Chapitre VIII gdal_fillnodata

Usage :

```
gdal_nodatafill [-q] [-md max_distance] [-si smooth_iterations]
            [-o name=value] [-b band]
            srcfile [-nomask] [-mask filename] [-of format] [dstfile]
```

Description :

Le script `gdal_nodatafill.py` remplit les régions sélectionné (généralement des zones *nodata*) en interpolant à partir de pixels valides autour des bords de la zone.

Des détails supplémentaires sur l'algorithme sont disponibles sur la documentation de *GDALFillNodata()*.

- **-q :** le script se lance en mode silencieux. La barre de progression est supprimée et les messages ne sont pas affichés.
- **-md max_distance :** la distance maximale (en pixels) que l'algorithme recherchera pour les valeurs à interpoler.
- **-si smooth_iterations :** le nombre d'itérations de filtre d'atténuation moyen 3x3 à lancer après l'interpolation pour amortir les artefacts. Par défaut il n’y a aucune itération d'atténuation.
- **-o name=value :** définie un argument spécial à l'algorithme. Pour l'instant non géré.
- **-b band :** la bande sur laquelle opérée, par défaut la première bande est utilisée.
- **srcfile :** le fichier source raster utilisé pour identifier les pixels cibles. Seule une bande est utilisée.
- **-nomask :** n'utilise pas le masque de validité par défaut pour la bande en entrée (tel que *nodata* ou le masque alpha).
- **-mask filename :** utilise la première bande du fichier spécifié comme masque de validité (zéro est invalide, autre que zéro est valide).
- **dstfile :** le nouveau fichier à créer avec les résultats interpolés. S'il n'est pas fourni, la bande source sera mise à jour à la place.
- **-of format :** sélectionne le format de sortie. Par défaut GeoTIFF (GTIFF). Utiliser le nom court du format.

Chapitre IX gdallocationinfo

Outil de requête raster

A Usage

Usage :

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

B Description

La commande `gdallocationinfo` fournie un mécanisme pour demander des informations sur un pixel pour une localisation donnée dans l'un des différents systèmes de coordonnées gérés. Plusieurs options de rapport sont proposés.

- **-xml :** le rapport en sortie sera formaté en XML pour facilité le post-traitement.
- **-lifonly :** le seul retour est la production des noms de fichier à partir de la demande de LocationInfo en fonction de la base de données (ie pour identifier le fichier impacté à partir des VRT).
- **-valonly :** le seul retour sont les valeurs du pixel sélectionné sur chaque bandes sélectionnées.
- **-b band :** sélectionne une bande à interroger. Plusieurs bandes peuvent être listées. Par défaut toutes les bandes sont interrogées.
- **-l_srs srs def :** Le système de coordonnées de la localisation x et y en entrée.
- **-geoloc :** indique que les points x et y en entrée sont dans le système de géoréférencement de l'image.
- **-wgs84 :** indique que les points x et y en entrée sont en lat, long WGS84.
- **srcfile :** le nom de la source de données raster GDAL.
- **x :** localisation X du pixel cible. Par défaut le système de coordonnées est pixel/ligne sauf si *-l_srs*, *-wgs84* ou *-geoloc* sont fournie.
- **y :** localisation Y du pixel cible. Par défaut le système de coordonnées est pixel/ligne sauf si *-l_srs*, *-wgs84* ou *-geoloc* sont fournie.

Cette commande a pour objectif de fournir diverses informations d'un pixel. Pour l'instant

il rapporte trois choses :

- la localisation du pixel dans l'espace pixel/ligne.
- le résultat d'une requête de méta-données LocationInfo en fonction de la source de données - pour l'instant cela est seulement implémenté pour les fichiers VRT qui rapportera le(s) fichier(s) utilisé(s) pour satisfaire les requêtes pour ce pixel.
- la valeur du pixel du raster pour ce pixel pour toutes ou un sous-ensemble de bande(s).
- la valeur du pixel non ajusté si une échelle et/ou un décalage est appliqué à la bande.

Le pixel sélectionné est interrogé par les coordonnées x/y sur la ligne de commande, ou lu par l'entrée standard (stdin). Plus d'une paire de coordonnées peut être fournies lors de la lecture des coordonnées par l'entrée standard. Par défaut, les coordonnées pixel/ligne sont attendue. Cependant lors de l'utilisation des options *-geoloc*, *-wgs84*, ou *-l_srs* il est possible de définir la localisation dans d'autres système de coordonnées.

Le rapport par défaut est au format texte lisible. Il est possible de récupérer une sortie en xml avec l'option *-xml*.

Afin de pouvoir utiliser des scripts, les options *-valonly* et *-lifonly* sont fournie pour restreindre la sortie aux valeurs de pixels réels, ou les fichiers LocationInfo identifiés pour le pixel.

Il est prévue que des possibilités de rapport supplémentaire seront ajouté à `gdallocationinfo` dans le futur.

C Exemple

Exemple simple reportant un pixel (256,256) sur le fichier *utm.tif*.

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

Requêter un fichier VRT fournissant la localisation en WGS84 et récupérant le résultat en 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>
```

Chapitre X gdal-config

Détermine diverses informations sur l'installation de GDAL.

Usage :

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

Ce script (disponible seulement sur les systèmes Unix) peut être utilisé pour déterminer les diverses informations sur l'installation de GDAL. Il est normalement juste utilisé par les scripts de configuration des applications utilisant GDAL mais il peut être invoqué par l'utilisateur final.

- **--prefix :** le niveau le plus haut pour le répertoire d'installation de GDAL.
- **--libs :** bibliothèques et liens nécessaires pour GDAL.
- **--cflags :** définitions d'include et de macro nécessaire pour compiler des modules en utilisant GDAL.
- **--version :** renvoi la version de GDAL.
- **--ogr-enabled :** renvoi "yes" ou "no" vers la sortie standard selon que OGG a compilé avec GDAL ou non.
- **--formats :** renvoi quels formats sont configurés dans GDAL vers la sortie.

Chapitre XI Les formats de fichiers GDAL

Chapitre XII AIRSAR -- AIRSAR Polarimetric Format

La plupart des variantes du format Polarimétrique d'AIRSAR produit par le Processeur Intégré d'AIRSAR sont gérés en lecture par GDAL. Les produits d'AIRSAR incluent normalement divers fichiers associés, mais seules les données d'image elles-mêmes sont gérées. Normalement celles-ci sont nommées *mission_l.dat* (Bande L) ou *mission_c.dat* (Bande C).

Le format AIRSAR contient une image polarimétrique sous forme de matrice de dispersion compressée. GDAL décompresse la donnée dans une matrice de stockage puis convertit cette forme dans une matrice de covariance. Les 6 bandes retournées sont les six valeurs nécessaire pour définir la matrice 3x3 de covariance d'Hermitian. La convention utilisée pour représenter la matrice de covariance en terme d'éléments de matrice de dispersion HH, HV (=VH) et VV est indiqué ci-dessous. Notez que les éléments non-diagonal de la matrice sont des valeurs complexes, tandis que les valeurs diagonales sont des réels (bien que représenté par des bandes complexes).

- Band 1 : Covariance_11 (Float32) = HH*conj(HH)
- Band 2 : Covariance_12 (CFloat32) = sqrt(2)*HH*conj(HV)
- Band 3 : Covariance_13 (CFloat32) = HH*conj(VV)
- Band 4 : Covariance_22 (Float32) = 2*HV*conj(HV)
- Band 5 : Covariance_23 (CFloat32) = sqrt(2)*HV*conj(VV)
- Band 6 : Covariance_33 (Float32) = VV*conj(VV)

L'identité des bandes sont également reflété dans les méta-données et les descriptions des bandes.

Le format du produit AIRSAR inclut (potentiellement) plusieurs en-tête d'information. Ces informations sont capturé et représenté comme des méta-données sur le fichier dans son ensemble. Les objets des informations à partir de l'en-tête principal sont préfixé de « MH_ », les objets de l'en-tête du paramètre sont préfixé de « PH_ » et les informations de l'en-tête de calibration sont préfixé de « CH_ ». Les noms des objets des méta-données sont dérivé automatiquement du nom des champs de l'en-tête lui-même.

Aucun effort n'est fait pour lire les fichiers associés avec le produit d'AIRSAR tel que *mission_l.mocomp*, *mission_meta.airsar* ou *mission_meta.podaac*.

Lisez également :

- **AIRSAR Data Format :** http://airsar.jpl.nasa.gov/documents/dataformat.htm

Chapitre XIII BLX -- Magellan BLX Topo File Format

BLX est le format pour le stockage de données topographique dans les GPS Magellan. Ce pilote gère à la fois la lecture et l'écriture. De plus les 4 niveaux d'aperçus inhérent au format BLX peut être utilisé avec le pilote.

Le format BLX est basé sur des tuiles pour le moment la taille des tuiles est fixée à une taille de 128x128. De plus les dimensions doivent être un multiple de la taille de la tuile.

Le type de données est fixé à Int6 et la valeur pour les valeurs non définie est définie à -32768. Dans le format BLX les valeurs non définies sont seulement gérées au niveau de la tuile. Pour les pixels non définie dans des tuiles non vides voyez les options **FILLUNDEF/FILLUNDEFVAL**.

A Géoréférencement

La projection BLX est fixée à WGS84 et le géoréférencement à partir des BLX est géré sous la forme de tiepoint et de taille de pixel.

B Problèmes lors de la création

Options de création :

- **ZSCALE=1 :** définie l'incrémentation de le quantisation désirée pour l'accès en écriture. Une valeur plus importante résultera en une meilleure compression et une moins bonne résolution verticale.
- **BIGENDIAN=YES :** si **BIGENDIAN** est définie, le fichier en sortie sera au format XLB (big endian blx).
- **FILLUNDEF=YES :** si **FILLUNDEF** est définie à **yes** la valeur de **FILLUNDEFVAL** sera utilisé à la place de -32768 pour les tuiles vides. Ceci est nécessaire puisque le format BLX ne gère que les valeurs indéfinie pour les tuiles complètes, pas pour les pixels individuels.
- **FILLUNDEFVAL=0 :** Voir FILLUNDEF

Chapitre XIV BAG --- Bathymetry Attributed Grid

Ce pilote permet la gestion en lecture seule des données bathymétriques au format BAG. Les fichiers BAG sont en fait un profile d'un produit spécifique dans un fichier HDF5, mais un pilote personnalisé existe pour présenter les données dans une manière plus pratique que celle disponible via le pilote HDF5 générique.

Les fichiers BAG ont deux ou trois bandes d'images représentant les valeurs de l'élévation (bande 1), l'incertitude (bande 2) et l'élévation nominale (bande 3) pour chaque cellule dans une zone en grille raster.

La transformation géographique et le système de coordonnées sont extraits des métadonnées XML interne fournie par le jeu de données. Cependant, certain produits peuvent avoir des formats du système de données non gérés.

Le XML complet de métadonnées est disponible dans le domaine de métadonnées "xml:BAG".

Les valeurs Nodata, minimum et maximum pour chaque bande sont également fournies.

A Voir également

- Implémenté dans *gdal/frmts/hdf5/bagdataset.cpp*.
- Le projet Open Navigation Surface <http://www.opennavsurf.org>`_

Chapitre XV BMP --- Microsoft Windows Device Independent Bitmap

Bitmaps Indépendant des Périphériques de MS Windows gérés par le noyau de Windows et le plus souvent utilisé pour stocker les images de décoration du système. Dû à la nature du format BMP il y a plusieurs restrictions et ne il peut pas être utilisé pour le stockage d'image générale. En particulier, vous pouvez seulement créer des images monochrome sur 1 bit, pseudo-couleur sur 8 bits et RVB de 24 bits. Même si des images en nuance de gris doivent être sauvé sous la forme pseudo-couleur.

Ce pilote gère la lecture de presque n'importe quel type de fichiers BMP et peut en écrire un qui devrait être géré sur n'importe quel système Windows. Seulement des fichiers simple- ou tri-bandes peut être sauvé dans un fichier BMP. Les valeurs en entrée seront re-échantillonnées en 8 bits.

Si un fichier worl ESRI exist avec l'extension .bpw, .bmpw ou .wld, il sera lu et utilisé pour établir la géotransformation pour l'image.

Options de création :

- **WORLDFILE=YES :** force la génération d'un fichier world d'ESRI associé (avec l'extension .wld).

Lisez également :

- Implémenté dans *gdal/frmts/bmp/bmpdataset.cpp*.
- **Référence Bitmap MSDN :** http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_9qg5.asp

Chapitre XVI COSAR -- TerraSAR-X Complex SAR Data Product

Ce pilote fournit la possibilité de lire les données complexes TerraSAR-X. Bien que la plupart des utilisateurs reçoivent les données au format GeoTIFF (représentant la radiation détectée réfléchit à partir des cibles, les produits ScanSAR seront distribué au format COSAR.

Pour l'essentiel, COSAR est une matrice binaire annotée, avec chaque échantillon contenu sous 4 bytes (16 bytes réel, 16 bytes imaginaire) stocké avec le byte le plus significatif en premier (Big Endian). Dans un conteneur COSAR il y a un ou plusieurs "bursts" qui représentent des bursts ScanSAR individuel. Notez que si un produit Stripmap ou Spotlight est contenu dans un conteneur COSAR il est stocké dans un seul burst.

La gestion des données ScanSAR est en cours de dévéloppement, dû à la difficulté de s'adapter aux identifiant des "bursts" ScanSAR dans le modèle GDAL.

Voir également :

- DLR Document TX-GS-DD-3307 "Level 1b Product Format Specification."

Chapitre XVII Formats divers

A AAIGrid -- Arc/Info ASCII Grid

Géré pour l'accès en lecture et écriture, incluant la lecture d'une transformation affine de géoréférencement et certaines projections. Ce format est le format ASCII d'échange pour Arc/Info Grid, et prennent la forme d'un fichier ASCII, plus parfois un fichier .prj associé. Il est normalement produit par la commande `ASCIIGRID` d'Arc/Info.

La gestion des projections (lu si un fichier *.prj est disponible) est assez limitée. Des exemples additionnels de fichier .prj peuvent être envoyé au mainteneur, warmerdam@pobox.com.

La valeur `NODATA` pour la lecture de la grille est également préservée lorsqu'elle est disponible.

Par défault, le type de données renvoyé pour les jeux de données AAIGRID par GDAL est auto-détecté, et définie à Float32 pour les grilles avec des valeurs de point en virgule flottante ou sinon en Int32. Cela est réalisé par l'analyse du format des valeurs NODATA et les premiers 1000 ko de données de la grille. À partir de GDAL 1.8.0, vous pouvez explicitement définir le type de données en définissant l'option de configuration *AAIGRID_DATATYPE* (les valeurs Int32, Float32 et Float64 sont actuellement gérés)

Si les pixels écrits ne sont pas carrés (la largeur et la hauteur du pixel dans l'unité géoréférencé diffèrent) alors les paramètres DX et DY seront affichés à la place de CELLSIZE. De tel fichier peuvent être utilisé dans Golden Surfer, mais pas dans la plupart des programmes de lecture de grille ASCII. Pour forcer la taille du pixel X à utiliser comme `CELLSIZE` utilisez l'option de création `FORCE_CELLSIZE=YES` ou re-échantillonné l'entrée pour avoir des pixels carrés.

Lors de l'écriture de valeurs en virgules flottantes, le pilote utilise le motif de format "*%6.20g* par défaut. Vous pouvez lire le manuel de référence pour printf pour avoir une idée du comportement exact de ceci ;-). vous pouvez aussi spécifier le nombre de décimal avec l'option de création `DECIMAL_PRECISION`. par exemple `DECIMAL_PRECISION=3` renverra des nombres avec trois décimales.

Le pilote AIG est également disponible pour le format de grille binaire d'Arc/Info.

Note!

Implémenté dans gdal/frmts/aaigrid/aaigriddataset.cpp.

B ACE2 -- ACE2

(à partir de GDAL >= 1.9.0)

C'est un pilote de commodité afin de lire les MNT d'ACE2 . Ces fichiers contiennent des données binaires brutes. Le géoréférencement est entièrement déterminé par le nom du fichier. Qualité, source et confiance des couches sont de type Int16, alors que les données d'altitude sont retournées comme Float32.

Aperçu du produit ACE2

Note!

Implémenté dans gdal/frmts/raw/ace2dataset.cpp.

C ADRG/ARC Digitized Raster Graphics (.gen/.thf)

Géré par GDAL en lecture. La création est possible mais doit être considérée comme expérimentale et une manière de tester l'accès en lecture (bien que les fichiers créés par le pilote peut être lu avec succès par d'autres logiciels).

Un jeu de données EADGR est composé de plusieurs fichiers. Le fichier reconnu par GDAL est le Fichier d'Information Général (.GEN). GDAL a besoin également du fichier image (.IMG) dans lequel se trouve les données.

Le fichier d'En-tête de Transmission (.THF) peut également être utilisé comme une entrée à GDAL. Si le THF référence plus d'une image, GDAL renverra la composition des images comme sous jeu de données. Si le THF référence une seule image seulement, GDAL l'ouvrira directement.

Les aperçues, légendes et insets ne sont pas utilisé. Les zones Polaires (ARC Zone 9 et 18) ne sont pas gérées (dû au manque de données tests).

C'est une alternative à l'utilisation de OGDI Bridge pour les jeux de données ADRG.

Voyez également : Les spécification ADRG (MIL-A-89007)

D AIG -- Arc/Info Binary Grid

Géré par GDAL pour l'accès en lecture. Ce format est le format binaire interne pour les grilles Arc/Info, et prend la forme de répertoire de niveau de couverture (NdT, si vous trouvez mieux ...) dans une base de données Arc/Info. Pour ouvrir la couverture sélectionnez le répertoire de la couverture, ou un fichier .adf (tel que hdr.adf) à l'intérieure. Si le répertoire ne contient pas de fichier(s) avec un nom comme *w001001.adf* alors ce n'est pas une couverture de grille.

La gestion inclut la lecture d'une transformation affine du géoréférencement, certaines projections et une table de couleur (.clr) si disponible.

Ce pilote a été développé sur la base d'un *reverse-engineering* du format. Lisez la description du format pour plus de détails : http://home.gdal.org/projects/aigrid/index.html.

La gestion des projections (lu si un fichier prj.adf est disponible) est assez limité. Des exemples additionnels de fichier .prj peuvent être envoyé au mainteneur, warmerdam@pobox.com.

Note!

Implémenté dans gdal/frmts/aigrid/aigdataset.cpp.

E Format BSB -- Maptech/NOAA BSB Nautical Chart

Le format BSB Nautical Chart est géré en accès en lecture, incluant la lecture de la table de couleur et les points de références (comme les points d'amer). Notez que les fichiers .BSB ne peuvent pas être sélectionné directement. À la place sélectionnez les fichiers .KAP. Les versions 1.1, 2.0 et 3.0 ont été testées avec succès.

Ce pilote doit également géré le format GEO/NOS tel que fournit par Softchart. Ces fichiers normalement ont l'extension .nos avec des fichiers .geo associés contenant le géoréférencement ... Les fichiers .geo sont pour l'instant ignoré. Ce pilote est basé sur le travail de Mike Higgins. Lisez les fichiers frmts/bsb/bsb_read.c pour les détails sur les brevets affectant le format BSB.

À partir de GDAL 1.6.0, il est possible de sélectionner une palette de couleur alternative via l'option de configuration `BSB_PALETTE`. La valeur par défaut est *RGB*. Voici d'autres valeurs qui peuvent être utilisées : *DAY*, *DSK*, *NGT*, *NGR*, *GRY*, *PRC*, *PRG* ...

Note!

Implémenté dans gdal/frmts/bsb/bsbdataset.cpp.

F Format BT -- VTP .bt Binary Terrain

Le format .BT est utilisé pour les donnés d'élévation dans le logiciel VTP. Le pilote inclut la gestion pour la lecture et l'écriture du format .bt 1.3 incluant la gestion des types de données des pixels en Int16, Int32 et Float32. Le pilote ne gère pas la lecture et l'écriture des fichiers compressés (.bt.gz) même si cela est géré par le logiciel VTP. S'il vous plaît, décompressez les fichiers avant d'utiliser GDAL avec "gzip -d file.bt.gz".

Les projections dans les fichiers .prj externes sont lu et écrit, et la gestion pour la plupart des systèmes de coordonnées définie en interne est également disponible.

L'accès des images en lecture et écriture avec le pilote .bt de GDAL est terriblement lent à cause de l'inefficacité de la stratégie d'accès aux colonnes de données. Cela pourrait être corrigé, mais demanderait un effort important.

Note!

Implémenté dans gdal/frmts/raw/btdataset.cpp.

Lisez également : Le format de fichier BT est défini sur le site Web de VTP : http://www.vterrain.org/Implementation/Formats/BT.html.

G CEOS -- CEOS Image

C'est un simple lecteur pour les fichiers images ceaos. Pour l'utiliser, sélectionné le fichier d'imagerie principale. Ce pilote lit seulement les données images, et ne récupère pas les méta-données ou le géoréférencement.

Ce pilot est connu pour fonctionner avec les données CEOS produites par Spot Image, mais présente des problèmes avec plusieurs autres sources de données. En particulier, il ne fonctionnera qu'avec les données non signées sous 8 bits.

Voyez le pilote séparé SAR_CEOS (page 75, E.XXXV.29) pour accéder aux produits de

données SAR CEOS.

Note!

Implémenté dans gdal/frmts/ceos/ceosdataset.cpp.

H DODS/OPeNDAP – lecture des rasters à partir de serveurs DODS/OPeNDAP

Gestion pour l'accès en lecture des serveurs DODS/OPeNDAP. Envoie l'URL DODS/OPeNDAP au pilote tel que vous l'aurez accéder pour un fichier local. L'URL définit le serveur distant, le jeu de données et les rasters dans le jeu de données. De plus, vous devez dire au pilote quelles dimensions doivent être interprétées comme bandes distinctes ainsi que laquelle correspond à la latitude et la longitude. Lisez le fichier README.DODS pour de plus amples informations.

I DOQ1 -- Première génération USGS DOQ

Gestion de l'accès en lecture, incluant la lecture d'une transformation du géoréférencement affine, et la capture de la projection. Ce format est le vieux format, non étiqueté DOQ (Digital Ortho Quad) de l'USGS.

Note!

Implémenté dans gdal/frmts/raw/doq1dataset.cpp.

J DOQ2 – Nouveau USGS DOQ étiqueté

Gestion pour l'accès en lecture, incluant la lecture d'une transformation du géoréférencement affine, et la capture de la projection et la lecture des autres champs auxiliaires comme métadonnées. Ce pilote est le nouveau format, étiqueté DOQ (Digital Ortho Quad) de l'USGS.

Ce pilote a été développé par Derrick J Brashear.

Note!

Implémenté dans gdal/frmts/raw/doq2dataset.cpp.

Lisez également : les standards DOQ de l'USGS sur http://rockyweb.cr.usgs.gov/nmpstds/doqstds.html

K E00GRID -- Arc/Info Export E00 GRID

(GDAL >= 1.9.0)

GDAL gère la lecture des raster/MNT exporté comme grilles E00.

Le pilote a été testé avec des jeux de données tels que ceux disponibles sur ftp://msdis.missouri.edu/pub/dem/24k/county/

Note!

Implémenté dans gdal/frmts/e00grid/e00griddataset.cpp.

L EHdr -- ESRI .hdr Labelled

GDAL gère la lecture et l'écriture du format d'étiquette .hdr d'ESRI, souvent appelé format BIL d'ESRI. Les types de données raster d'entier en 8, 16 et 32 bits sont gérés ainsi que les virgules flottantes en 32 bites. Les systèmes de coordonnées (à partir d'un fichier .prj) et le géoréférencement sont gérés. Les options non reconnues dans le fichier .hdr sont ignorées. Pour ouvrir un jeu de données, sélectionnez le fichier avec le fichier image (souvent avec l'extension .bil). Si présent, le fichier des tableaux de couleurs .clr sont lu mais pas écrit.

Ce pilote ne fait pas toujours la différenciation entre les données en virgules flottantes et en entier. L'extension GDAL au format .hdr pour les différencier est d'ajouter un champ nommé *PIXELTYPE* avec des valeurs parmi *FLOAT*, *SIGNEDINT* ou *UNSIGNEDINT*. En combinaison avec le champ *NBITS* il est possible de décrire toutes les variations des types de pixel.

Par exemple :

```
ncols 1375
nrows 649
cellsize 0.050401
xllcorner -130.128639
yllcorner 20.166799
nodata_value 9999.000000
nbits 32
pixeltype float
byteorder msbfirst
```

Ce pilote peut être suffisant pour lire les données GTOPO30.

Note!

Implémenté dans gdal/frmts/raw/ehdrdataset.cpp.

Lisez également :

- ESRI whitepaper : Formats d'image étendue pour ArcView GIS 3.1 et 3.2 (BIL, voir p. 5) : http://downloads.esri.com/support/whitepapers/other_/eximgav.pdf
- GTOPO30 - Global Topographic Data : http://edcdaac.usgs.gov/gtopo30/gtopo30.html
- Documentation sur GTOPO30 : http://edcdaac.usgs.gov/gtopo30/README.html
- :ref:`gdal.gdal.formats.divers_formats.srtmhgt`
  Unknown interpreted text role "ref".

M ENVI - ENVI .hdr Labelled Raster

GDAL gère certaines variations de fichiers raster brute avec un fichier.hdr de styles ENVI associés décrivant le format. Pour sélectionner un fichier raster ENVI existant sélectionnez le fichier binaire contenant la donnée (par opposition aux fichier .hdr), et GDAL trouvera le fichier .hdr en remplaçant l'extension du jeu de données par .hdr.

GDAL devrait gérer la lecture des formats bil, bip et bsq interlacée, et la plupart des types de pixel sont gérés, incluant les entiers sur 8 bit non signés, 16 et 32 bits signés et non signés, les virgules flottantes sur 32 et 64 bits et les virgules flottantes complexes sur 32 et 64 bits. Il y a une gestion limitée pour la reconnaissance du mot-clé map_info avec le système de coordonnées et le géoréférencement. En particulier, UTM et State Plane devraient fonctionner.

Options de création :

- `INTERLEAVE=BSQ/BIP/BIL` : force la génération d'un type définie d'interlacement. BSQ --- band sequental (par défaut), BIP --- data interleaved by pixel, BIL --- data interleaved by line.
- `SUFFIX=REPLACE/ADD` : force l'ajout du suffixe ".hdr" au fichier fournit, par exemple, si l'utilisateur sélectionne le nom "file.bin" pour le nom en sortie du jeu de données, le fichier d'en-tête "file.bin.hdr" sera crée. Par défaut le suffixe du fichier d'en-tête remplace le suffixe du fichier binaire, par exemple pour "file.bin" le fichier d'en-tête nommé "file.hdr" sera créé.

Note!

Implémenté dans gdal/frmts/raw/envidataset.cpp.

N Envisat -- Envisat Image Product

GDAL gère le format du produit Envisat en accès en lecture. Tous les types d'échantillon sont gérés. Les fichiers avec deux jeux de données de mesures correspondantes (MDS) sont représentés comme ayant deux bandes. Pour l'instant tous les produits ASAR de niveau 1 et supérieur et quelques produits MERIS et AATSR sont gérés.

Les points de contrôles des jeux de données GEOLOCATION GRID ADS sont lus si elles sont disponibles, généralement en donnant une bonne couverture du jeu de données. Les points d'amer sont en WGS84.

Virtuellement toutes les paires clés/valeurs du MPH et SPH (en-têtes Primaire et Secondaire) sont copiées comme des métas-données de niveau du jeu de données.

Les paramètres ASAR et MERIS contenue dans les enregistrements ADS et GADS (sauf ceux de la géolocalisation) peuvent être récupérés sous forme de pair de clé/valeur en utilisant le domaine de métadonnées "RECORDS".

Note!

Implémenté dans gdal/frmts/envisat/envisatdataset.cpp.

Lisez également : Envisat Data Products à l'ESA : http://envisat.esa.int/dataproducts/

O FITS -- Flexible Image Transport System

FITS est un format utilisé principalement par les astronomes, mais c'est un format relativement simple qui gère les types d'images arbitraires et les images multispectrales et donc a trouvé son utilisation dans GDAL. La gestion de FITS est implémentée par la bibliothèque SFITSIO standard (http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html) que vous devez avoir sur votre système dans le but d'activer la gestion FITS. À la fois la lecture et l'écriture de fichiers FITS sont gérées. À ce moment, aucune gestion pour un système de géoréférencement n'est développée, mais la gestion du WCS (World Coordinate System) est possible dans le futur.

Les mots-clés d'en-tête non standard qui sont présents dans le fichier FITS seront copiés vers les méta-données du jeu de données quand le fichier est ouvert, pour l'accès par les méthodes de GDAL. De même, les mots-clés non standard que l'utilisateur définit dans les méta-données du jeu de données seront écrits dans le fichier FITS quand la prise en charge de GDAL sera fermée.

Remarque à ceux qui sont familiers avec la bibliothèque `CFITSIO` : la regraduation

automatique des valeurs des données, déclenchée par la présence des mots-clés d'en-tête `BSCALE` et `BZERO` dans un fichier FITS, est désactivée dans GDAL. Ces mots-clés d'en-tête sont accessible et peuvent être mise à jour par les méta-données du jeu de données, de la même manière que les autres mots-clés d'en-tête, mais ils n'affectent pas la lecture/l'écriture des valeurs des données à partir de/vers le fichier.

Note!

Implémenté dans gdal/frmts/fits/fitsdataset.cpp.

P GRASSASCIIGrid -- GRASS ASCII Grid

(GDAL >= 1.9.0)

Gère la lecture du format grille ASCII de GRASS (similaire à la commande ASCIIGRID d'Arc/Info).

Part défaut, le type des données renvoyé pour les jeux de données grilles ASCII de GRASS par GDAL est autodétecté, et définie à Float32 pour les grilles avec des valeurs en virgules flottantes ou sinon Int32. Cela est réalisé par l'analyse du format des valeurs nulles et les premiers 100 ko de onnées de la grille. Vous pouvez aussi explicitement définir le type de données en définissant l'option de configuration *GRASSASCIIGRID_DATATYPE* (les valeurs Int32, Float32 et Float64 sont géré pour l'instant).

Note!

Implémenté dans gdal/frmts/aaigrid/aaigriddataset.cpp.

Q GSAG -- Golden Software ASCII Grid File Format

C'est la version basé sur l'ASCII (lisible par un être humain) d'un des formats raster utilisé par les produits de Golden Software (tels que ceux de la série Surfer). Ce format est géré à la fois en lecture et en écriture (création, suppression et copie incluse). Pour l'instant les formats associés pour la couleur, les méta-données, et les formes ne sont pas gérés.

Note!

Implémenté dans gdal/frmts/gsg/gsagdataset.cpp.

R GSBG -- Golden Software Binary Grid File Format

C'est la version binaire (non lisible par un être humain) d'un des formats raster utilisés par les produits de Golden Software (tels que ceux de la série Surfer). Comme pour la version ASCII, ce format est géré à la fois en lecture et en écriture (création, suppression et copie inclus). Pour l'instant les formats associés pour la couleur, les méta-données, et les formes ne sont pas gérés.

Note!

Implémenté dans gdal/frmts/gsg/gsbgdataset.cpp.

S GS7BG -- Golden Software Surfer 7 Binary Grid File Format

C'est la version binaire (non lisible par un être humain) d'un des formats raster utilisés par les produits de Golden Software (tels que ceux de la série Surfer). Ce format diffère du format GSBG (connu également comme le format grille binaire de Surfer 6), il est plus compliqué et moins flexible. Ce format est géré en lecture seule.

Note!

Implémenté dans gdal/frmts/gsg/gs7bgdataset.cpp.

T GXF -- Grid eXchange File

C'est un format d'échange de raster diffusé par Geosoft, et en fait un standard dans le champ de la gravité/magnétique. Le format est géré en lecture et écriture et inclus la gestion des informations de géo-référencement et de projections.

Par défaut, le type de données renvoyé pour les jeux de données GXF par GDAL est Float32. À partir de GDAL 1.8.0, vous pouvez définir le type de données en définissant l'option de configuration *GXF_DATATYPE* (Float64 géré pour le moment)

Détails sur le code géré, et le format peuvent être trouvé sur la page GXF-3 http://home.gdal.org/projects/gxf/index.html

Note!

Implémenté dans gdal/frmts/gxf/gxfdataset.cpp.

U IDA -- Analyse et affichage d'image

GDAL gère la lecture et l'écriture des images IDA avec quelques limitations. Les images IDA sont les images du format de WinDisp 4. Les fichiers ont toujours une bande de données 8 bits. Les fichiers IDA ont souvent l'extension .img bien que cela n'est pas requis.

Les informations de projection et de géoréférencement est lu bien que certaines projections (c'est à dire Météosat et Hammer-Aitoff) ne sont pas gérés. Lors de l'écriture des fichiers IDA la projection doit avoir un false easting et false northing de zéro. Les systèmes de coordonnées gérés dans les fichiers IDA sont Géographique, Lambert Conformal Conic, Lambert Azimuth Equal Area, Albers Equal-Area Conic et Goodes Homolosine.

Les fichiers IDA contiennent typiquement des valeurs échantillonnées en 8 bits via une pente et un décalage. Ceux-ci sont retournés comme les valeurs de pente et de décalage de la bande et ils doivent être utilisés si la donnée doit être re-échantillonée vers les valeurs brutes originales pour analyse.

Note!

Implémenté dans gdal/frmts/raw/idadataset.cpp.

Lisez également : WinDisp : http://www.fao.org/giews/english/windisp/windisp.htm

V JDEM -- Japanese DEM (.mem)

GDAL inclut la gestion de la lecture pour les fichiers DEM Japonais, ayant normalement

l'extension .mem. Ces fichiers sont un produit de la Japanese Geographic Survey Institute.

Ces fichiers sont représentés par une bande d'entiers flottants de 32bit avec des données d'élévation. Le géoréférencement des fichiers est retourné ainsi que le système de coordonnées (toujours en Lat/Lon sur le datum de Tokyo). Il n'y a pas de gestion de la mise à jour ou de la création pour ce format.

Note!

Implémenté dans gdal/frmts/jdem/jdemdataset.cpp.

Lisez également : Le site Web de Geographic Survey Institute (GSI) : http://www.gsi.go.jp/ENGLISH/

W LAN -- Erdas 7.x .LAN et .GIS

GDAL gère la lecture des fichiers raster Erdas 7.x .LAN et GIS. Pour l'instant les types de données des pixels de 4 bits, 8 bits et 16 bits sont gérés pour la lecture et de 8 et 16 bits pour l'écriture.

GDAL lit l'étendue des cartes (geotransform) à partir des fichiers LAN/GIS, et tente de lire les informations du système de coordonnées. Cependant, ce format de fichier n'inclut pas complètement les informations du système de coordonnées, donc pour les systèmes de coordonnées UTM et state plane une définition de LOCAL_CS est renvoyé avec des unités linéaires valides, mais aucune autres informations significatives.

Les fichiers .TRL, .PRO et world sont ignorés pour le moment.

Note!

Implémenté dans gdal/frmts/raw/landataset.cpp

Le développement de ce pilote a été financé par Kevin Flanders de PeopleGIS (http://www.peoplegis.com/).

X MFF -- Vexcel MFF Raster

GDAL inclut la gestion de la lecture, la mise à jour et la création du format raster MFF de Vexcel. Les jeux de données MFF consistent en un fichier d'en-tête (typiquement avec l'extension .hdr) et un ensemble de fichiers donnés avec des extensions comme .x00. .b00 etc. Pour ouvrir un jeu de donné sélectionnez le fichier .hdr.

La lecture des points d'amer Lat/Lon (TOP_LEFT_CORNER, ...) est gérée, mais il n'y a pas de gestion pour la lecture des informations de projections ou de transformation affine.

Les mots-clé non reconnus du fichier .hdr sont préservés comme méta-données.

Tous les types de données avec un équivalents GDAL sont gérés, incluant les précisions des types de données entiers, réels et complexes en 8, 16, 32 et 64 bites. De plus, les fichiers organisés en tuile (comme produit par le Vexcel SAR Processor – APP) sont gérés en lecture.

En création (avec un code de format de MFF) un fichier raster simple et non géoréférencé est créé.

Les fichiers MFF ne sont pas normalement portables entre les systèmes avec différents ordres d'octets. Cependant, GDAL utilise le nouveau mot-clé BYTE_ORDER qui peut

prendre la valeur de LSB (Integer -- little endian), et MSB (Motorola -- big endian). Cela peut être manuellement ajouté au fichier .hdr si nécessaire.

Note!

Implémenté dans gdal/frmts/raw/mffdataset.cpp.

Y NDF -- NLAPS Data Format

GDAL a une gestion limitée des fichiers de Format de Données NLAPS. C'est un format d'abord utilisé par le Centre de Données Eros pour la distribution des données Landsat. Les jeux de données NDF contiennent un fichier d'en-tête (souvent avec une extension .Hl) et un ou plus de fichiers de données brutes associées (souvent .I1, .I2, ...). Pour ouvrir un jeu de données sélectionner le fichier d'en-tête, souvent avec l'extension.H1, .H2 ou .HD.

Le pilote NDF gère seulement les données 8 bises. La seule projection gérée est UTM. La version 1 de NDF (NDF_VERSION=0.00) et la version 2 de NDF sont toutes deux gérées.

Note!

Implémenté dans gdal/frmts/raw/ndfdataset.cpp.

Lisez également : Les spécifications du format de Données NLAPS sur la page http://landsat.usgs.gov/documents/NLAPSII.pdf

Z GMT -- GMT Compatible netCDF

GDAL a une gestion limitée pour la lecture et l'écriture des fichies grid de netCDF. Les fichiers netCDF qui ne sont pas reconnus comme grilles (il manque des variables appelées dimension et z) seront ignorés silencieusement par ce pilote. Ce pilote a d'abord l'objectif de fournir un mécanisme pour l'échange de grille avec le paquet GMT (http://gmt.soest.hawaii.edu/). Le pilote netCDF doit être utilisé pour des jeux de données betCDF plus générales.

L'information des unités dans le fichier sera ignoré, mais les informations x_range, et y_range seront lut pour obtenir les éténdus de géoréférencement du raster. Tous les types de données netCDF doivent être gérés en lecture. Les fichiers nouvellement crées (avec un type de GMT) auront toujours comme unité le mètre pour x, y et z mais les valeurs de x_range, y_range et z_range doivent être correct. Remarquez que netCDF n'ont pas de type de données non signé en byte, les rasters 8 bites devront être généralement convertis en Int16 pour l'exporter vers GMT.

La gestion de netCDF dans GDAL est optionnelle et n'est pas compilée par défaut.

Note!

Implémenté dans gdal/frmts/netcdf/gmtdataset.cpp.

Lisez également : Unidata NetCDF Page : http://www.unidata.ucar.edu/packages/netcdf/

AA PAux -- PCI .aux Labelled Raw Format

GDAL inclut un développement partiel des fichiers rasters brutes étiquetées .aux pour la lecture, l'écriture et la création. Pour ouvrir un fichier étiquetté PCI, sélectionné le fichier de données brutes lui-même. le fichier .aux (qui doit avoir un nom identique) sera

utilisé automatiquement.

Le type de format pour la création de nouveaux fichiers est PAux. Tous les types de données (8U, 16U, 16S, et 32R) sont gérés. Pour l'instant, le géo-référencement, les projections et les autres méta-données sont ignorés.

1 Options de création

- **INTERLEAVE=PIXEL/LINE/BAND :** établit l'entrelacement de la sortie, BAND par défaut.

Note!

Implémenté dans gdal/frmts/raw/pauxdataset.cpp.

Voyez également : Description du format .aux de PCI

AB PCRaster raster file format

GDAL inclut la gestion de la lecture et l'écriture de fichiers raster PCRaster. PCRaster est un système de modélisation dynamique pour des modèles de simulation distribués. Les principales applications de PCRaster se trouvent dans la modélisation environnementale : géographie, hydrologie, écologie pour en nommer quelques-uns. Des exemples incluent des modèles d'écoulement des eaux de pluie, modèles de compétition de la végétation et des modèles de stabilité des pentes.

Le pilote lit tous les types de cartes PCIRaster : booléens, nominales, ordinales, scalaire, directionnel et ldd. La même représentation de la cellule utilisée pour stocker les valeurs dans le fichier est utilisée pour stocker les valeurs en mémoire.

Le pilote détecte si la source du raster GDAL est un fichier PCRaster. Quand un tel raster est écrit dans un fichier de l'échelle de valeur du raster originel sera utilisé. Le pilote écrit **toujours** les valeurs en utilisant des représentations de la cellule UINT1, INT4 or REAL4, en fonction de l'échelle de valeurs :

Échelle de valeurs	***Représentation de la cellule***
VS_BOOLEAN	CR_UINT1
VS_NOMINAL	CR_INT4
VS_ORDINAL	CR_INT4
VS_SCALAR	CR_REAL4
VS_DIRECTION	CR_REAL4
VS_LDD	CR_UINT1

Pour les rasters d'autres sources qu'un fichier PCRaster une échelle de valeurs et une représentation de la cellule sont déterminées en fonction des règles suivantes :

Type de la source	***Échelle de valeur cible***	***Représentation cible de la cellule***
GDT_Byte	VS_BOOLEAN	CR_UINT1
GDT_Int32	VS_NOMINAL	CR_INT4

GDT_Float32	VS_SCALAR	CR_REAL4
GDT_Float64	VS_SCALAR	CR_REAL4

Le pilote peut convertir les valeurs d'une représentation de cellule gérée à un autre. Il ne peut pas convertir vers des représentations de cellule non gérée. Par exemple, il n'est pas possible d'écrire un fichier raster PCIRaster à partir de valeurs qui sont utilisées comme CR_INT2 (GDT_Int16).

Bien que l'extension de fichier raster PCRaster soit de facto *.map*, le logiciel PCRaster ne nécessite pas une extension de fichier standard.

Note!

Implémenté dans gdal/frmts/pcraster/pcrasterdataset.cpp.

Lisez également : PCRaster website at Utrecht University et PCRaster Environmental Software company website.

AC PNG -- Portable Network Graphics

GDAL inclut une gestion de la lecture et de la création des fichiers .png. Les fichiers en nuance de gris, pseudo-couleur, avec une palette, RVB et RVBA sont gérés ainsi que les précisions de 8 et 16 bits par échantillon.

Les fichiers PNG sont linéairement compressés, la lectuer aléatoire de gros fichier PNG peut être inefficace (résultat de plusieurs redémarrages de la décompression à partir du début du fichier).

Les textes importants sont traduits en méta-données, typiquement avec des lignes multiples par objet. Les :ref:`gdal.gdal.formats.divers_formats.wld` avec les extensions .pgw, .pngw ou .wld seront lu. Les valeurs de transparence simple dans les fichiers en nuance de gris seront reconnues comme des valeurs *nodata* dans GDAL. Les index de transparence dans les images avec palette sont préservés quand la table de couleur est lu.

Unknown interpreted text role "ref".

Les fichiers PNG peuvent être crée avec un type de PNG, en utilisant la méthode `CreateCopy()`, nécessitant un prototype que l'on peut lire. L'écriture inclus la gestion pour divers types d'images, et préservera les valeurs nodata/transparence. Les fichiers de géoréférencement .wld sont écrit si l'option WORLDFILE est définie. Tous les types de pixels autres que 16 bite non signés seront écrit sous huit bites.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraites du fichier, et seront stockés comme contenu brute XML dans le domaine de métadonnées xml:XMP.

Options de création :

- **WORLDFILE=YES :** force la génération d'un fichier world ESRI associé (avec l'extension .wld). Lisez la section fichier World WLD -- ESRI World File, pour plus de détails.
- **ZLEVEL=n :** définie la quantité de temps à utiliser pour la compression. La valeur par défaut est 6. Une valeur de 1 est rapide mais ne compresse pas, et une valeur de 9 est lent mais compresse beaucoup mieux.

Note!

Implémenté dans gdal/frmts/png/pngdataset.cpp.

La gestion de PNG a été développée sur la base de la bibliothèque de référence libpng. Plus d'information est disponible sur http://www.libpng.org/pub/png.

AD PNM -- Netpbm (.pgm, .ppm)

GDAL inclut la gestion en lecture, et création des fichiers compatibles .pgm (nuance de gris), et .ppm (couleur RVB) avec l'outil Netpbm. Seul le format binaire (brute) est géré.

Les fichiers Netpbm peuvent être créés avec le type PNM.

Options de création :

- `MAXVAL=n` : force le paramétrage de la valeur maximale de la couleur à n dans le fichier PNM en sortie. Cela peut être utile si vous planifiez l'utilisation du fichier en sortie avec des logiciels qui ne sont pas libéraux à cette valeur.

Note!

Implémenté dans gdal/frmts/raw/pnmdataset.cpp.

AE Raster Product Format/RPF (a.toc)

C'est un lecteur (et seulement en lecture) de produits RPF, comme CADRG ou CIB qui utilise un fichier de contenu - *A.TOC* - à partir d'un échange RPF, et l'expose comme jeu de données virtuel dont la couverture est l'ensemble des cadres contenu dans la table de contenu.

Le pilote rapportera un sous jeu de données différents pour chaque sous jeu de données trouvé dans le fichier *A.TOC*.

Résultat d'une commande `gdalinfo` sur un fichier *A.TOC*.

```
Subdatasets:
    SUBDATASET_1_NAME=NITF_TOC_ENTRY:CADRG_GNC_5M_1_1:GNCJNCN/rpf/a.to
c
    SUBDATASET_1_DESC=CADRG:GNC:Global Navigation Chart:5M:1:1
[...]
    SUBDATASET_5_NAME=NITF_TOC_ENTRY:CADRG_GNC_5M_7_5:GNCJNCN/rpf/a.to
c
    SUBDATASET_5_DESC=CADRG:GNC:Global Navigation Chart:5M:7:5
    SUBDATASET_6_NAME=NITF_TOC_ENTRY:CADRG_JNC_2M_1_6:GNCJNCN/rpf/a.to
c
    SUBDATASET_6_DESC=CADRG:JNC:Jet Navigation Chart:2M:1:6
[...]
    SUBDATASET_13_NAME=NITF_TOC_ENTRY:CADRG_JNC_2M_8_13:GNCJNCN/rpf/a.
toc
    SUBDATASET_13_DESC=CADRG:JNC:Jet Navigation Chart:2M:8:13
```

Dans certaines situations, les tuiles NITF (voir :ref:`gdal.gdal.formats.nitf`) dans le sous-jeu de données ne partagent pas la même palette. Le pilote RPFTOC fera du mieux qu'il peut pour recartographier les palettes à la palette rapportée par `gdalinfo` (qui est la palette de la première tuile du sous jeu de données). Dans les situations où il ne donnerait pas de bon résultat, vous pouvez tenter de définir la variable d'environnement `RPFTOC_FORCE_RGBA` à `TRUE` avant l'ouverture du sous-jeu de données. Cela entraînera l'exposition du sous-jeu de données RVBA par le pilote au lieu d'un jeu avec une palette.

Unknown interpreted text role "ref".

Il est possible de construire les aperçus externes pour un sous jeu de données. L'aperçu pour le premier sous-jeu de données sera nommé *A.TOC.1.ovr* par exemple, pour le second jeu de données il sera nommé *A.TOC.2.ovr*, etc. Notez que vous devrez rouvrir le sous-jeu de données avec la même définition de `RPFTOC_FORCE_RGBA` que celui que vous avez utilisé lors de la création. N'utilisez pas une méthode autre que le ré-échantillonnage de NEAREST lors de la construction des aperçus sur un sous-jeu de données avec palette (RPFTOC_FORCE_RGBA non définie).

Une commande `gdalinfo` sur un de ces sous jeu de données retournera les différentes méta-données NITF ainsi que la liste des tuiles NITF du sous-jeu de données.

Voir également :

- Pont OGDI : le pilote RPFTOC propose des fonctionnalités équivalentes (sans les dépendances externes) au pilote RPF de la bibliothèque OGDI.
- MIL-PRF-89038 : spécifications de RPF, CADRG, CIB

Note!

Implémenté dans gdal/frmts/nitf/rpftocdataset.cpp

AF SAR_CEOS -- CEOS SAR Image

C'est un lecteur en lecture seul pour les fichiers images CEOS SAR. Pour l'utiliser, sélectionner le fichier image principal. Ce pilote fonctionne avec la plupart des produits de données Radarsat et ERS, incluant les produits complexes ; cependant, il est improbable qu'il fonctionne pour les produits autres que Radar CEOS. Le pilote CEOS plus simple est souvent approprié pour ceux-ci (http://www.remotesensing.org/gdal/frmt_various.html#CEOS). Le pilote tentera de lire 15 points d'amer lat/long en échantillonnant l'information de la superstructure de CEOS par ligne. Il capture également divers méta-données à partir de divers fichiers d'en-tête, incluant :

```
CEOS_LOGICAL_VOLUME_ID=EERS-1-SAR-MLD
CEOS_PROCESSING_FACILITY=APP
CEOS_PROCESSING_AGENCY=CCRS
CEOS_PROCESSING_COUNTRY=CANADA
CEOS_SOFTWARE_ID=APP 1.62
CEOS_ACQUISITION_TIME=19911029162818919
CEOS_SENSOR_CLOCK_ANGLE=  90.000
CEOS_ELLIPSOID=IUGG_75
CEOS_SEMI_MAJOR=    6378.1400000
CEOS_SEMI_MINOR=    6356.7550000
```

Le pilote SAR_CEOS inclut également certaines gestions pour les données polarimétriques SIR-C et PALSAR. Le format SIR-C contient un image sous forme de matrice de dispersion compressée, décrit ici http://southport.jpl.nasa.gov/software/dcomp/dcomp.html. GDAL décompresse la donnée au moment de la lecture. Le format PALSAR contient des bandes qui correspondent presque exactement aux éléments d'une matrice de covariance d'Hermitian de 3x3- Lisez le document ERSDAC-VX-CEOS-004A.pdf sur http://www.ersdac.or.jp/palsar/palsar_E.html pour une description complète (stockage des pixels est décrit à la page 193). GDAL convertit celles-ci en bandes de matrices de covariance de point flottant complexe au fur et à mesure qu'ils sont lus. La convention utilisée pour représenter la matrice de covariance en terme d'éléments de matrice de dispersion HH, HV (=VH) et VV est indiquée ci-dessous. Notez que les éléments non

diagonaux de la matrice sont des valeurs complexes, tandis que les valeurs diagonales sont des réels (bien que représenté par des bandes complexes).

- Band 1 : Covariance_11 (Float32) = HH*conj(HH)
- Band 2 : Covariance_12 (CFloat32) = sqrt(2)*HH*conj(HV)
- Band 3 : Covariance_13 (CFloat32) = HH*conj(VV)
- Band 4 : Covariance_22 (Float32) = 2*HV*conj(HV)
- Band 5 : Covariance_23 (CFloat32) = sqrt(2)*HV*conj(VV)
- Band 6 : Covariance_33 (Float32) = VV*conj(VV)

L'identité des bandes est également reflétée dans les métas-données.

Note!

Implémenté dans gdal/frmts/ceos2/sar_ceosdataset.cpp.

AG CTG -- USGS LULC Composite Theme Grid

(GDAL >= 1.9.0)

Ce pilote peut lire les grilles *Land Use and Land Cover* (LULC) de l'USGS encodées au format *Character Composite Theme Grid* (CTG). Chaque fichier est renvoyé comme un jeu de données à 6 bandes de type Int32. La signification de chaque bande est celui-ci :

1. Code d'utilisation et de couvertures des sols (*Land Use and Land Cover Code*) ;
2. Code des unités politiques (*Political units Code*) ;
3. Code des subdivisions de recensement du comté et de tracts SMSA (*Census county subdivisions and SMSA tracts Code*) ;
4. Codes des unités hydrologiques (*Hydrologic units Code*) ;
5. Code des propriétaires du sol Fédéral (*Federal land ownership Code*) ;
6. Code de propriété du sol de l'état (*State land ownership Code*) ;

Ces fichiers sont typiquement nommés grid_cell.gz, grid_cell1.gz ou grid_cell2.gz sur le site USGS.

- Land Use and Land Cover Digital Data (Data Users Guide 4) - version PDF de l'USGS
- Duplicate explicit target name: "land use and land cover digital data (data users guide 4)".
  Land Use and Land Cover Digital Data (Data Users Guide 4) - version HTML convertie par Ben Discoe ;
- Données LULC de l'USGS à 250K et 100K

Note!

Implémenté dans gdal/frmts/ctg/ctgdataset.cpp.

AH DIMAP -- Spot DIMAP

C'est un pilote en lecture seul pour les images décrites Spot DIMAP. Pour l'utiliser, sélectionnez le fichier METADATA.DIM dans le répertoire du produit, ou le répertoire même du produit.

L'image est un fichier image distinct, souvent un fichier TIFF, mais le jeu de données DIMAP prend en charge l'accès à ce fichier, et attache la géolocation et d'autres méta-données au jeu de données à partir du fichier XML de méta-données.

À partir de GDAL 1.6.0, le contenu des noeuds "Spectral_Band_Info" est renvoyé comme méta-données au niveau de la bande raster. Notez que le contenu de *Spectral_Band_Info*

de la première bande est encore renvoyé comme méta-données du jeu de données, mais cela doit être considéré comme un moyen déprécié d'obtenir cette information.

Note!

implémenté dans gdal/frmts/dimap/dimapdataset.cpp.

AI SDAT -- SAGA GIS Binary Grid File Format

(à partir de GDAL 1.7.0)

Le pilote gère la lecture et l'écriture (dont la création, la suppression et la copie) de grille binaire de SAGA GIS. Les jeux de données grille binaire de SAGA sont faite de fichier d'en-tête ASCII (.SGRD) et de données binaires (.SDAT) avec un nom de fichier commun. Le fichier .SDAT doit être sélectionné pour accéder au jeu de données.

Le pilote gère la lecture des types de données de SAGA suivantes (entre parenthèse les types GDAL correspondantes) : BIT (GDT_Byte), BYTE_UNSIGNED (GDT_Byte), BYTE (GDT_Byte), SHORTINT_UNSIGNED (GDT_UInt16), SHORTINT (GDT_Int16), INTEGER_UNSIGNED (GDT_UInt32), INTEGER (GDT_Int32), FLOAT (GDT_Float32) et DOUBLE (GDT_Float64).

Le pilote gère l'écriture des types de données SAGA suivantes : BYTE_UNSIGNED (GDT_Byte), SHORTINT_UNSIGNED (GDT_UInt16), SHORTINT (GDT_Int16), INTEGER_UNSIGNED (GDT_UInt32), INTEGER (GDT_Int32), FLOAT (GDT_Float32) et DOUBLE (GDT_Float64).

Pour le moment le pilote ne gère pas le zFactors autre que 1 et la lecture des grilles SAGA qui ont été écrite TOPTOBOTTOM.

Note!

Implémenté dans gdal/frmts/saga/sagadataset.cpp.

AJ SDTS -- USGS SDTS DEM

GDAL inclut la gestion de la lecture des DEM formatés en USGS SDTS. Les fichiers DEM de l'USGS sont toujours renvoyé avec un type de données entier de 16 bite non signé, ou un flottant de 32 bit. Les informations de géoréférencement et de projection sont aussi renvoyées.

Les jeux de données SDTS consistent en un certain nombre de fichiers. Chaque DEM doit avoir un fichier avec un nom comme XXXCATD.DDF. Celui-ci doit être sélectionné pour ouvrir le jeu de données.

Les unités d'élévation d'un DEM peuvent être les mestres ou les pieds. La méthode GetType() sur un bande tentera de retourner si les unités sont des peis (« ft ») ou des mètres (« m »).

Note!

implémenté dans gdal/frmts/sdts/sdtsdataset.cpp.

AK SGI - SGI Image Format

Le pilote SGI gère pour l'instant la lecture et l'écriture des fichiers images SIG.

Le pilote gère aujourd'hui les images à 1, 2, 3 et 4 bandes. Il gère les images de « 8 bites

par canal de valeur » et les images à la fois non compressées et run-length encoded (RLE) en lecture, mais les fichiers créés ont toujours une compression RLE.

Le pilote SGI de GDAL était basé sur le code de lecture d'image SGI de Paul Bourke.

Lisez également :

- Code de lecture des images SGIS de Paul Bourke : http://astronomy.swin.edu.au/~pbourke/dataformats/sgirgb/
- Document sur le format des fichiers images SGI : ftp://ftp.sgi.com/graphics/SGIIMAGESPEC

Note!

Implémenté dans gdal/frmts/sgi/sgidataset.cpp.

AL SNODAS -- Snow Data Assimilation System

(À partir de GDAL >= 1.9.0)

C'est un pilote commodité pour lire les données Snow Data Assimilation System. Ces fichiers contiennent des données binaires brutes en Int16. Le fichier à fournir à GDAL est le fichier.Hdr.

Produits de données Snow Data Assimilation System (SNODAS) à NSIDC

Note!

Implémenté dans gdal/frmts/raw/snodasdataset.cpp.

AM Standard Product Format (ASRP/USRP) (.gen)

(à partir de GDAL 1.7.0)

Les produits ASRP et USRP (comme définie par la DGIWG) sont des variations sur des formats de produits standards plus comment et sont gérés en lecture par GDAL. Les jeux de données ASRP et USRP sont fait de plusieurs fichiers - typiquement de fichiers .GEN, .IMG, .SOU et .QAL avec un nom de fichier commun. Le fichier .GEN doit être sélectionné pour accéder au jeu de données.

Les produits ASRP (dans un système de coordonnées géographiques) et USRP (dans un système de coordonnées UTM/UPS) sont des images à une seule bande avec une palette et un géoéréférencement.

Note!

Implémenté dans gdal/frmts/adrg/srpdataset.cpp

AN SRTMHGT - SRTM HGT Format

Le pilote SRTM HGT gère aujourd'hui la lecture des fichiers SRTM-3 et SRTM-1 V2 (HGT).

Le pilote gère la création des nouveaux fichiers, mais les données en entrée doivent être exactement formatées en cell SRTM-3 ou SRTM-1. C'est-à-dire que la taille, et les limites doivent être appropriées pour une cellule.

Lisez également :

- SRTM documentation

- SRTM FAQ 1
- SRTM FAQ 2

Note!

Implémenté dans gdal/frmts/srtmhgt/srtmhgtdataset.cpp.

AO ECRG Table Of Contents (TOC.xml)

À partir de GDAL 1.9.0

C'est un lecteur en lecture seule pour les produits ECRG (Enhanced Compressed Raster Graphic), qui utilise le fichier de table de contenu, TOC.xml, et l'expose comme jeu de données virtuel dont la couverture est l'ensemble de cadre ACRG contenu dans la table de contenu.

Le pilote renverra un sous jeu de données différent pour chaque sous jeu de données trouvés dans le fichier TOC.xml.

Résultat de la commande `gdalinfo` sur un fichier TOC.xml :

```
Subdatasets:
SUBDATASET_1_NAME=ECRG_TOC_ENTRY:ECRG:FalconView:ECRG_Sample/EPF/TOC.x
ml
SUBDATASET_1_DESC=ECRG:FalconView
```

1 Voir également

- :ref:`gdal.gdal.formats.nitf` : format des cadres ECRG ;
  Unknown interpreted text role "ref".
- MIL-PRF-32283 : spécification des produits ECRG.

Note!

Implémenté dans gdal/frmts/nitf/ecrgtocdataset.cpp.

AP EIR -- Erdas Imagine Raw

GDAL gère le format Erdas Imagine Raw pour l'accès en lecture incluant les entiers non signés 1, 2, 4, 8, 16 et 32 bit, les entiers signés 16 et 32 bit et les virgules flottantes complexes 32 et 64 bits. Le géoréférencement est géré.

Pour ouvrir un jeu de données, sélectionner le fichier avec les informations d'en-tête. Le pilote trouve le fichier image à partir des informations d'en-tête. Les documents Erdas appelle le fichier en-tête du fichier brut et il peut avoir l'extension .raw bien que le fichier image qui contient les données brutes réels peuvent avoir l'extension .bl.

Note : Implémenté dans *gdal/frmts/raw/eirdataset.cpp*

AQ WLD -- ESRI World File

Un fichier world file est un fichier texte ASCII consistant à 6 valeurs séparées par des nouvelles lignes. Le format est :

```
pixel X size
rotation about the Y axis (usually 0.0)
```

```
rotation about the X axis (usually 0.0)
negative pixel Y size
X coordinate of upper left pixel center
Y coordinate of upper left pixel center
```

Par exemple :

```
60.0000000000
0.0000000000
0.0000000000
-60.0000000000
440750.0000000000
3751290.0000000000
```

Vous pouvez construire ce fichier simplement en utilisant votre éditeur de texte favori.

Les fichiers world file habituellement ont un suffixe .wld, ou parfois .tfw, tifw, .jgw ou d'autres suffixes en fonction du fichier image avec lequel il est fournit.

AR XPM - X11 Pixmap

GDAL inclut la gestion pour la lecture et l'écriture des fichiers image XPM (Format Pixmap X11). Ceux-ci sont des images à une bande de cartes de couleur d'abord utilisé à de simples but graphiques dans les applications X11. Il a été incorporé dans GDAL d'abord pour faciliter la traduction des images GDAL en une forme utilisable avec le toolkit GTK.

La gestion du XPM ne gère pas le géoréférencement (non disponible à partir des fichiers XPM) ni ne gère les fichiers XPM avec plus d'un caractère par pixel. Les nouveaux fichiers XPM doivent avoir une carte de couleur ou être en nuance de gris, et les tables de couleurs seront réduites à 70 couleurs automatiquement.

Note!

Implémenté dans gdal/frmts/xpm/xpmdataset.cpp.

AS GenBin - Binaire Générique (étiqueté .hdr)

Ce pilote gère la lecture des fichiers "Binaire Générique" étiquetés avec un fichier .hdr mais distinct du format plus commun d'ESRI étiqueté .hdr (pilote EHdr). L'origine de ce format n'est pas très claire. Les fichiers .hdr gérés par ce pilote ressemble à cela :

```
{{{
BANDS:       1
ROWS:     6542
COLS:     9340
...
}}}
```

Les types de données U8, U16, S16, F32, F64, et U1 (bit) des pixels sont gérés. Le géoréférencement et les informations du système de coordonnées devraient être gérés lorsqu'ils sont fournis.

Note!

Implémenté dans gdal/frmts/raw/genbindataset.cpp

AT GFF - Sandia National Laboratories GSAT File Format

Le pilote GDAL en lecture seul a été pensé pour fournir un accès aux données traitées à partir des différents capteurs expérimentaux des Laboratoires Nationale de Sandia. Le format est essentiellement un en-tête de longueur arbitraire contenant la configuration des instruments et les paramètres de performances en fonction d'une matrice binaire de données complexes de 16 ou 32 bits ou de bytes réels.

Le format GFF a été implémenté sur la base du code Matlab fourni par Sandia pour lire les données. Le pilote gère tous les types de données (complexe sur 16 ou 32 bits, bytes réels) théoriquement, cependant dû à un manque de données seules les données complexes sur 32 bits ont été testées.

Sandia fournit des données échantillon à http://sandia.gov/RADAR/sar-data.html.

L'extension pour les formats GFF est .gff.

Note!

Implémenté dans gdal/frmts/gff/gff_dataset.cpp.

AU ZMap -- ZMap Plus Grid

(à partir de GDAL >= 1.9.0)

Géré pour l'accès en lecture et la création.

Ce format est un format d'échange ASCII pour les données en grille dans un format en ligne ASCII pour le transport et le stockage. Il est communément utilisé dans les applications dans la champs d'Exploration Pétrolière et Gazière.

Par défaut, les fichiers sont interprétés et écrit en fonction de la convention PIXEL_IS_AREA. Si vous définissez l'option de configuration *ZMAP_PIXEL_IS_POINT* à TRUE, la convention *PIXEL_IS_POINT* sera suivie pour interpréter/écrire le fichier (les valeurs géoréférencées dans l'en-tête du fichier seront alors considérées comme les coordonnées du centre des pixels). Notez que dans ce cas, GDAL renverra l'étendue avec sa convention usuelle PIXEL_IS_AREA (les coordonnées du coin haut à gauche comme reporté par GDAL sera une moitié de pixel en haut et à gauche des valeurs qui apparaît dans le fichier).

Spécification informelle donnée dans le thread de la liste de diffusion GDAL-dev

Note!

Implémenté dans gdal/frmts/zmap/zmapdataset.cpp.

Chapitre XVIII DODS -- OPeNDAP Grid Client

GDAL inclut en option la gestion en lecture des grilles 2D et des tableaux via le protocole OPeNDAP (DODS).

A dénomination des jeux de données

La spécification complète du nom des jeux de données consiste en une URL du jeu de données OPeNDAP, le chemin complet vers le tableau désiré ou la variable grille, et un indicateur des indices du tableau à accéder.

Par exemple, si l'url http://maps.gdal.org/daac-bin/nph-hdf/3B42.HDF.dds renvoie une définition DDS comme celle-ci :

```
Dataset {
Structure {
    Structure {
    Float64 percipitate[scan = 5][longitude = 360][latitude = 80];
    Float64 relError[scan = 5][longitude = 360][latitude = 80];
    } PlanetaryGrid;
} DATA_GRANULE;
} 3B42.HDF;
```

alors la grille peut être accéder en utilisant le nom du jeu de données suivant dans GDAL :

http://maps.gdal.org/daac-bin/nph-hdf/3B42.HDF?
DATA_GRANULE.PlanetaryGrid.percipitate[0][x][y]

Le chemin complet vers la grille ou le tableau à accéder nécessite d'être définie (sans compter le nom du jeu de données extérieur). GDAL doit savoir quels indices du tableau correspond au x (longitude ou coordonnées à l'est). Toutes les autres dimensions nécessite d'être limité à une valeur seule.

Dans le cas de serveurs de données avec seulement des tableaux 2D et des grilles comme enfant immédiat du jeu de données il peut ne pas être nécessaire de nommer grille ou la variable tableau.

Dans les cas où il y a un grand nombre de tableaux 2D ou de grilles au niveau du jeu de données, ils peuvent être chacun automatiquement traité comme bandes séparées.

B Méta-données spécialisées AIS/DAS

Diverses informations seront transportées via le DAS décrivant le jeu de données. Certains pilotes DODS (tels que celui basé sur GDAL) retourne déjà les informations DAS suivantes mais dans d'autres cas il peut être fournie localement en utilisant le mécanisme AIX. Lisez la documentation sur DODS pour les détails du fonctionnement du mécanisme AIS.

```
Attributes {

    GLOBAL {
        Float64 Northernmost_Northing 71.1722;
        Float64 Southernmost_Northing  4.8278;
        Float64     Easternmost_Easting  -27.8897;
        Float64     Westernmost_Easting -112.11;
        Float64 GeoTransform "71.1722 0.001 0.0 -112.11 0.0 -0.001";
        String spatial_ref "GEOGCS[\"WGS
84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
84\",6378137,298.257223563]],PRIMEM[\"Greenwich\",0],UNIT[\"degree\",0
.0174532925199433]]";
        Metadata {
        String TIFFTAG_XRESOLUTION "400";
        String TIFFTAG_YRESOLUTION "400";
        String TIFFTAG_RESOLUTIONUNIT "2 (pixels/inch)";
        }
    }

    band_1 {
        String Description "...";
        String
    }
}
```

1 Jeu de données

Il y aura un objet dans le DAS nommé GLOBAL contenant les attributs du jeu de données dans l'ensemble.

Il possède les sous-items suivants :

- **Northernmost_Northing :** La latitude ou la valeur la plus au nord du bord au nord de l'image.
- **Southernmost_Northing :** La latitude ou la valeur la plus au nord du bord au sud de l'image.
- **Easternmost_Easting :** La longitude ou la valeur la plus à l'est du bord à l'ouest de l'image.
- **Westernmost_Easting :** La longitude ou la valeur la plus à l'est du bord à l'est de l'image.
- **GeoTransform :** Les six paramètres définissant la transformation affine entre les pixels/l'espacement des lignes et l'espace géoréférencé si applicable. Stocké comme un chaîne de caractère simple avec des valeurs séparées par des espaces. Notez que cela permet des images en rotation ou inclinées (optionnel)

- **SpatialRef :** La descriptions OpenGIS WKT du système de coordonnées. Si absente, il sera supposé que le système de coordonnées est le WGS84. (optionnel)
- **Metadata :** un conteneur avec une liste d'attributs de chaînes de caractères pour chaque item de méta-données disponible. Le nom du mot-clé de l'item de la méta-données sera utilisé comme nom d'attribut. Les valeurs des méta-données seront toujours nue chaîne. (optionnel)
- address GCPs

Notez que les valeurs des bords nord et est peuvent être calculé à partir de la taille de la grille et de la transformation géométrique. Ils sont d'abord inclus comme documentation supplémentaire qui est plus facile à interpréter par l'utilisateur que la transformation géométrique. Ils seront également utilisé pour calculer une transformation géométrique interne si l'une d'elle n'est pas fournit, mais si les deux sont fournit la transformation géométrique prendra le dessus.

2 Bande

Il y aura un objet dans le DAS nommé après chaque bande contenant des attributs d'une bande spécifique.

Il aura les sous-items suivants :

- **Metadata :** un conteneur avec une liste d'attributs de chaîne pour chaque item de méta-données disponilbe. Le nom du mot-clés de l'item de la méta-données sera utilisé comme nom d'attribut. Les valeurs des méta-données seront toujours des chaînes. (optionnel)
- **PhotometricInterpretation :** aura une valeur parmi "Undefined", "GrayIndex", "PaletteIndex", "Red", "Green", "Blue", "Alpha", "Hue", "Saturation", "Lightness", "Cyan", "Magenta", "Yellow" ou "Black". (optionnel)
- **units :** nom des unités (parmi "ft" ou "m" pour les données d'élévation). (optionnel)
- **add_offset :** déplacement à appliquer aux valeur des pixels. (après scale_factor) pour calculer une valeur de pixel "réelle". Par défaut à 0.0. (optionnel)
- **scale_factor :** Redimensionnement à appliquer aux valeur du pixel (avant add_offset) pour calculer une valeur de pixel "réelle". Par défauts à 1.0. (optionnel)
- **Description :** Texte descriptif sur la bande. (optionnel)
- **missing_value :** La valeur *nodata* pour le raster. (optionnel)
- **Colormap :** Un conteneur avec un sous-conteneur pour chaque couleur dans la table de couleur, ressemblant au suivant. le composant alpha est optionnel et supposé à 255 (opaque) s'il est absent.

```
Colormap {
  Color_0 {
    Byte red 0;
    Byte green 0;
    Byte blue 0;
    Byte alpha 255;
  }
  Color_1 {
    Byte red 255;
    Byte green 255;
    Byte blue 255;
    Byte alpha 255;
  }
```

```
   . . .
}
```

Lisez également :

- Site OPeNDAP

Chapitre XIX DTED -- Military Elevation Data

GDAL gère les données d'élévation DTED de niveau 0, 1 et 2 en accès en lecture. Les données d'élévation sont retournées comme des entiers signés sur 16 bits. Une projection appropriée (toujours WGS84) et des informations de géo-référencement sont également retourné. Une diversité de champs d'en-tête sont renvoyé comme méta-données de niveau du jeu de données.

A Problèmes de lecture

1 Vitesse de lecture

Les données d'élévation dans les fichiers DTED sont organisé par colonne. Cette organisation des données ne fonctionne pas très bien avec les algorithmes de scan par ligne et peut causer des ralentissements, spécialement pour les jeux de données DTED de niveau 2. En définissant *GDAL_DTED_SINGLE_BLOCK=TRUE*, un jeu de données DTED complet sera considéré comme un bloc simple. Le premier accès au fichier sera lente, mais un nouvel accès sera beaucoup plus rapide. Utiliser seulement cette option si vous avez besoin de réaliser un calcul sur le fichier complet.

2 Problèmes de géoréférencement

La spécification DTED (MIL-PRF-89020B) affirme que le datum horizontal doit être le système Géodésique Mondiale (World Geodetic System, WGS 84). Cependant, il y a encore des gens pour utiliser d'ancien fichier de données géoréférencé en WGS 72. Un champ d'en-tête indique le code du datum horizontal, nous pouvons donc détecter et prendre en compte cette situation.

- si le datum horizontal définie dans le fichier DTED est le WGS 84 le pilote DTED représentera le WGS 84 comme SRS ;
- si le datum horizontal définie dans le fichier DTED est le WGS 72, le pilote DTED représentera le WGS 72 comme SRS et retrounera un problème.
- si le datum horizontal définie dans le fichier DTED est ni le WGS 84 ni le WGS 72, le pilote DTED représentera le WGS 84 comme SRS et enverra un message d'erreur.

3 Problèmes de checksum

le comportement par défaut du pilote DTED est d'ignorer le checksum lors de la lecture des données à parti du fichier. Cependant, vous pouvez définir la variable d'environnement DTED_VERIFY_CHECKSUM=YES si vous voulez que le checksum soit vérifié. Dans certain cas, le checksum écrit dans le fichier DTED est incorrecte (le producteur de données a mal fait son travail). Cela sera retourné comme un message important. Si le checksum écrit dans le fichier DTED et le checksum calculé à partir des données ne correspond pas, une erreur sera retournée.

B Problème lors de la création

Le pilote DTED doit gérer les nouveaux fichiers, mais les données en entrée doivent être formaté exactement comme une cellule de niveau 0, 1 ou 2. C'est à dire que la taille et les limites doivent être appropriées pour la cellule.

Lisez également :

- Implémenté dans *gdal/frmts/dted/dteddataset.cpp*.

Chapitre XX Le format ECW - ERDAS Compress Wavelets

GDAL supporte le format .ecw en lecture et en écriture. L'implémentation actuelle lit n'importe quel nombre de bande mais renvoi seulement des images en 8 bit. Les systèmes de coordonnées et les transformations du géo-référencement sont lu, mais dans certains cas les systèmes de coordonnées ne seront pas traduit.

Le support du pilote ECW dans GDAL est optionnel, et nécessite le lien avec la bibliothèque SDK ECW externe fournit par ERDAS.

Le kit de compression gratuit (Licence de type public) ECW supporte la compression des images jusqu'à 500 Mo. Pour compresser de très grosses images, il est nécessaire d'avoir une licence de la technologie ECW d'ERMapper. Les fichiers à compresser au format ECW doivent également être d'au moins 128 × 128. Les sources qui ne sont pas en 8 bites seront re-échantillonnées par la bibliothèque SDK ECW d'une manière un peu incompréhensible. Le résultat est une image en 8 bites.

Lors de l'écriture des informations du système de coordonnées dans les fichiers ECW, la plupart des systèmes de coordonnées courants ne sont pas intégrés proprement. Si vous connaissez le nom du système de coordonnées au format ERMapper, vous pouvez l'obliger à le définir au moment de la création avec les options de création du PROJ et du DATUM.

A Problèmes de création

En plus des fichiers ECW, ce pilote gère aussi l'accès au service de réseau d'image en utilisant le protocole "ECWP". Utilisez l'url complète *ecwp://* du service comme nom de jeu de données. Lorsqu'il compilé avec le SDK 4.1 ou plus récent il est aussi possible de tirer avantage des accès asynchrone des services ECWP de la RFC 24.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraite à partir des fichiers JPEG2000, et seront stockés comme contenu brute XML dans le domaine de métadonnées xml:XMP.

Options de création :

- **TARGET=pourcentage :** définit la taille de réduction cible comme un pourcentage de l'originale. S'il n'est pas définit, la valeur par défaut est de 90

pour les images en nuances de gris et de 95 pour les images RVB.

- **PROJ=nom :** Nom au format ECW de la chaîne de projection à utiliser. Des exemples courants sont NUTM11, ou GEODETIC.
- **DATUM=nom :** nom du datum au format d'ECW à utiliser. Des exemples courants sont WGS84 ou NAD83.
- **LARGE_OK=YES :** lorsque compilé avec le SDK ECW 3.x cette option peut être définie pour autoriser la compression des fichiers supérieur à 500 Mo. Il est de la responsabilité de l'utilisateur de s'assurer que les nécessités de licence pour la compression des gros fichiers a été suivit.
- **ECW_ENCODE_KEY=key :** fournie la clé d'encodage OEM acheté chez Erdas qui permet l'encodage des images. La clé a une longueur approximative de 129 hex. Il peut aussi être fournie comme option de configuration.
- **ECW_ENCODE_COMPANY=name :** fournie le nom de la société qui possède la clé d'encodage OEM d'ERDAS (voir ECW_ENCODE_KEY). Cela doit correspondre exactement au nom utilisé par ERDAS lors de la fourniture de la clé OEM. Il peut aussi être fournie comme option de configuration.

Le format ECW ne supporte pas la création d'aperçu puisque le format ECW est déjà censé être optimisé pour les « aperçues arbitraires ».

B Options de configuration

Le SDK ECW d'ERDAS gère une grande variété d'options de configuration pour contrôler différentes fonctionnalités. La plupart de celles-ci sont exposée par les options de configuration de GDAL. Voyez la documentation du SDK ECW pour plus de détails sur la signification de ces options.

- **ECW_CACHE_MAXMEM=bytes :** octets maximal de RAM utilisé pour la mise en cache mémoire. S'il n'est pas définie, jusqu'à 1/4 de la RAM physique sera utilisé par le SDK pour la mise en cache en mémoire.
- SDK ECW disponible sur www.ermapper.com.
- **ECWP_CACHE_LOCATION=path :** chemin vers le répertoire à utiliser pour la mise en cache des résultats de ECWP. Si non définie, la mise en ache ECWP ne sera pas activée.
- **ECWP_CACHE_SIZE_MB=number_of_megabytes :** le nombre maximal de Mo d'espace dans ECWP_CACHE_LOCATION à utilisé pour la mise en cache des résultats ECWP.
- **ECWP_BLOCKING_TIME_MS :** temps de lecture qu'un ecwp:// bloquera avant de revenir - 10 000 ms par défaut.
- **ECWP_REFRESH_TIME_MS :** délais entre l'arrivé des blocs et la prochaine demande de rafraîchissement - 10 000 ms par défaut. Dans le cas de GDAL ceci est le temps que le pilote attendra pour plus de données d'une connexion ecwp pour laquelle le résultat finale n'a pas été renvoyé. Si définie trop petit alors les requêtes *RasterIO()* produiront souvent des résultats de faibles

Bullet list ends without a blank line; unexpected unindent.

résolutions.

Block quote ends without a blank line; unexpected unindent.

- **ECW_TEXTURE_DITHER=TRUE/FALSE :** cela peut être définie à FALSE pour désactiver le tramage lors de la décompression des fichiers ECW. TRUE par défaut.
- **ECW_FORCE_FILE_REOPEN=TRUE/FALSE :** cela peut être définie à TRUE pour forcer à ouvrir un fichier pris en charge pour chaque fichier pour chaque

connexion réalisée. FALSE par défaut.

- **ECW_CACHE_MAXOPEN=number :** le nombre maximal de fichier à garder ouvert pour que le fichier ECW prenne en charge la mise en cache. Illimité par défaut.
- **ECW_RESILIENT_DECODING=TRUE/FALSE :** contrôle si le lecteur doit oublier les erreurs dans un fichier et essayer de renvoyer autant de données que possible. TRUE par défaut. Si définie à FALSE un fichier invalide résultera en une erreur.
- **ECW_ENCODE_KEY, ECW_ENCODE_COMPANY :** ces valeurs, comme décrite dans les options de création, peuvent aussi être définir comme options de configuration. voir plus haut.

C Voyez également

- Implementé dans *gdal/frmts/ecw/*.
- La page ECW (http://www.erdas.com/products/ERDASECWJPEG2000SDK/Details.aspx) chez http://www.erdas.com.
- Astuces de compilation de l'ECW pour GDAL

Chapitre XXI ELAS - Earth Resources Laboratory Applications Software

ELAS est una ancien système de capteur distant utilisé dans une variété de projets de recherche à la NASA. La gestion du format ELAS peut être trouvé dans `gdal/frmts/elas`.

Lisez également :

- Courte annonce du pilote ELAS dans GDAL
- Logiciels de la NASA utilisé dans les services d'imagerie avec un aperçu d'ELAS

Chapitre XXII Epsilon - images compressé par ondelette

À partir de GDAL 1.7.0, GDAL peut lire et écrire des images compressé par ondelette via la bibliothèque Epsilon.

À partir de GDAL 1.9.0, epsilon 0.9.1 est nécessaire.

Le pilote repose sur la bibliothèque Open Source EPSILON (double licence LGPL/GPL v3). Dans son état actuel, le pilote sera seulement capable de lire les images avec une tuilage interne régulier.

Le pilote EPSILON gère seulement les images d'1 bande (nuance de gris) et de 3 bandes (RVB).

Cela est censé être principalement utilisé par le pilote :ref:`gdal.gdal.formats.rasterlite`.

Unknown interpreted text role "ref".

A Options de création

- **TARGET** réduction cible de la taille comme pourcentage de l'original (0-100). 96 par défaut.
- **FILTER**. voir la documentation EPSILON ou `gdalinfo --format EPSILON` pour une liste complète des ID des filtres. 'daub97lift' par défaut.
- **BLOCKXSIZE=n :** définie la largeur des tuiles, 256 par défaut. Puissance de deux entre 32 et 1024
- **BLOCKYSIZE>=n :** définie la hauteur des tuiles, 256 par défaut. Puissance de deux entre 32 et 1024.
- **MODE=[NORMAL/OTLPF] :** OTLPF est une sorte de hack pour réduire les artefacts des contours quand l'image est découpée en plusieurs tuiles. Dû à des contraintes mathématiques cette méthode peut être être appliquée seulement aux filtres biorthogonaux. OTLPF par défaut.
- **RGB_RESAMPLE=[YES/NO] :** si le buffer RVB doit être re-échantillonné en 4:2:0. YES par défaut.

Voir également :

- Page principale d'EPSILON

Chapitre XXIII ERS -- ERMapper .ERS

GDAL gère la lecture et l'écriture de fichiers raster avec des fichiers d'en-têtes .ers avec certaines limitations. Le format ascii .ers est utilisé par ERMapper pour étiqueter les fichiers de données brutes ainsi que fournir des méta-données étendues et un géoréférencement pour certains autres fichiers. Le format .ERS et ses variantes sont également utilisés pour contenir les descriptions d'algorithmes d'ERMapper mais ceux-ci ne sont pas gérés par GDAL.

Voir également :

- Implémenté dans *gdal/frmts/ers/ersdataset.cpp*.

Chapitre XXIV FAST -- EOSAT FAST Format

Lecture gérée à partir des formats FAST-L7A (Landsat TM data) et EOSAT Fast Format Rev. C (IRS-1C/1D data). Si vous voulez lire d'autre jeu de de données dans ce format (SPOT), écrivez-moi (Andrey Kiselev, dron@ak4719.spb.edu). Vous pouvez partager des échantillons de données avec moi.

Les jeux de données au format FAST sont représentés par plusieurs fichiers : un ou plusieurs en-têtes administratifs et un ou plusieurs fichiers avec les véritables données des images dans un format brute. Les fichiers administratifs contiennent différentes informations sur les paramètres de scène incluant les noms des fichiers des images. Vous pouvez lire les fichiers avec les en-têtes administratifs avec n'importe quel éditeur de texte, ce sont juste des fichiers textes ASCII.

Ce pilote recherche les fichiers administratifs pour les fichiers en entrée. Les noms des fichiers des images seront extraits et les données seront importés, chaque fichier sera interprété comme une bande.

A Les données

1 FAST-L7A

FAST-L7A consiste en plusieurs fichiers : un gros avec les données de l'image et trois petits fichiers avec des informations d'administration. Vous devez donner au pilote un des fichiers d'administration :

- **L7fppprrr_rrrYYYYMMDD_HPN.FST :** ficher d'en-tête des bandes panchromatique avec 1 bande
- **L7fppprrr_rrrYYYYMMDD_HRF.FST :** fichier d'en-tête des bandes VNIR/ SWIR avec 6 bandes
- **L7fppprrr_rrrYYYYMMDD_HTM.FST :** fichier d'en-tête des bandes thermal avec 2 bandes

Toutes les images brutes et leur fichiers administratif correspondants seront importés comme des bandes GDAL.

À partir du document « Level 1 Product Output Files Data Format Control Book » (http://ltpwww.gsfc.nasa.gov/IAS/pdfs/DFCB_V5_B2_R4.pdf) :

```
The file naming convention for the FAST-L7A product files is
L7fpppprrr_rrrYYYYMMDD_AAA.FST

where
L7 = Landsat 7 mission
f = ETM+ format (1 or 2) (data not pertaining to a specific format
defaults to 1)
ppp = starting path of the product
rrr_rrr = starting and ending rows of the product
YYYYMMDD = acquisition date of the image
AAA = file type:
HPN = panchromatic band header file
HRF = VNIR/ SWIR bands header file
HTM = thermal bands header file
B10 = band 1
B20 = band 2
B30 = band 3
B40 = band 4
B50 = band 5
B61 = band 6L
B62 = band 6H
B70 = band 7
B80 = band 8
FST = FAST file extension
```

Vous devez donc donner au pilote un des fichiers parmi L7fpppprrr_rrrYYYYMMDD_HPN.FST, L7fpppprrr_rrrYYYYMMDD_HRF.FST ou L7fpppprrr_rrrYYYYMMDD_HTM.FST.

2 IRS-1C/1D

Le format Fast Rev. C ne contient pas les noms des fichiers des bandes dans les en-têtes administratifs. Nous devons donc deviner les noms des fichiers des bandes, parce que les différents distributeurs nomment leurs fichiers différemment. Plusieurs schémas de nommage sont codé dans le pilote FAST de GDAL. Ceux-ci sont :

```
<header>.<ext>
<header>.1.<ext>
<header>.2.<ext>
...
```

ou

```
<header>.<ext>
band1.<ext>
band2.<ext>
...
```

ou

```
<header>.<ext>
band1.dat
band2.dat
```

```
...
```

ou

```
<header>.<ext>
imagery1.<ext>
imagery2.<ext>
...
```

ou

```
<header>.<ext>
imagery1.dat
imagery2.dat
...
```

en majuscule ou minuscule. Le fichiers d'en-tête peut être nommé arbitrairement. Cela devrait couvrir la fantaisie de nommage des fichiers de la majorité des distributeurs. mais si vous n'avez pas de chance et votre jeu de données est nommé différemment vous devez les renommer manuellement avant l'importation des données avec GDAL.

B Géoréférencement

Toutes les projections USGS doivent être gérées (pour ne pas les nommer UTM, LCC, PS, PC, TM, OM, SOM). Contactez-moi si vous avez des problèmes avec l'extraction de la projection appropriée.

1 Méta-données

Les coefficients de calibration pour chaque bande signalée comme des objets de méta-données.

- ACQUISITION_DATE : date d'acquisition de la première scène au format yyyyddmm.
- SATELLITE : nom du satellite de la première scène.
- SENSOR : nom du capteur de la première scène.
- BIASn : valeur du biais pour le canal n.
- GAINn : valeur du gain pour le canal n.

Voir aussi :

- Implémenté dans *gdal/frmts/fast/fastdataset.cpp*.
- Description du format Landsat FAST L7A est disponible sur http://ltpwww.gsfc.nasa.gov/IAS/htmls/l7_review.html (Lisez ESDIS Level 1 Product Generation System (LPGS) Output Files DFCB, Vol. 5, Book 2 disponible sur http://ltpwww.gsfc.nasa.gov/IAS/pdfs/DFCB_V5_B2_R4.pdf)
- Description du format EOSAT Fast Format REV. C disponible sur http://www.euromap.de/docs/doc_001.html

Chapitre XXV GeoRaster d'Oracle Spatial

Ce pilote gère la lecture et l'écriture de données raster au format GeoRaster d'Oracle Spatial (10g ou supérieur). Le pilote GeoRaster d'Oracle Spatial est compilé en option comme plugin GDAL mais il nécessite les bibliothèques clientes d'Oracle.

Lors de l'ouverture du GeoRaster, son nom doit être définie sous la forme :

```
georaster:<user>{,/}<pwd>{,@}[db],[schema.][table],[column],[where]
georaster:<user>{,/}<pwd>{,@}[db],<rdt>,<rid>
```

où :

- *user* = login du nom d'utilisateur du serveur Oracle ;
- *pwd* = mot de passe utilisateur ;
- *db* = identification du serveur Oracle (nom de la base de données) ;
- *schema* = nom du schéma ;
- *table* = nom d'une table GeoRaster (table qui contient des colonnes GeoRaster) ;
- *column* = nom de la colonne données de type *MDSYS.SDO_GEORASTER* ;
- *where* = un clause WHERE simple pour identifier un ou plusieurs GeoRaster ;
- *rdt* = nom d'une table de données raster ;
- *rid* = identification numérique d'un GeoRaster.

Exemples :

```
geor:scott,tiger,demodb,table,column,id=1
geor:scott,tiger,demodb,table,column,"id = 1"
"georaster:scott/tiger@demodb,table,column,gain>10"
"georaster:scott/tiger@demodb,table,column,city='Brasilia'"
georaster:scott,tiger,,rdt_10$,10
geor:scott/tiger,,rdt_10$,10
```

Note!

N'utilisez pas d'espace autour des valeurs du champ et la virgule.

Note!

Comme dans les deux exemples précédents, le champs du nom de la base de données peut être laissé vide (",,") et le TNSNAME

sera utilisé.

Note!

Si la requête résulte en plus d'un GeoRaster elle sera traitée comme une liste de métadonnées de sous-jeu de données (voir plus bas).

A Naviguer dans la base de données des GeoRasters

En fournissant certaine information basique le pilote GeoRaster est capable de lister les rasters existant stockés sur le serveur :

Pour lister toutes les tables sur le serveur qui appartiennent à un utilisateur et à une base de données :

```
% gdalinfo georaster:scott/tiger@db1
```

Pour lister toutes les colonnes de types GeoRaster qui existent dans cette table :

```
% gdalinfo georaster:scott/tiger@db1,table_name
```

Ceci listera tous les objets GeoRaster stocké dans cette table :

```
% gdalinfo georaster:scott/tiger@db1,table_name,georaster_column
```

Ceci listera tous les GeoRaster existant sur cette table selon la clause QHERE :

```
% gdalinfo
georaster:scott/tiger@db1,table_name,georaster_column,city='Brasilia'
```

Notez que le résultat de ces requêtes est renvoyé à GDAL comme méta-données d'un sous jeu de données, par exemple :

```
% gdalinfo georaster:scott/tiger
Driver: GeoRaster/Oracle Spatial GeoRaster
Subdatasets:
SUBDATASET_1_NAME=georaster:scott,tiger,,LANDSAT
SUBDATASET_1_DESC=Table:LANDSAT
SUBDATASET_2_NAME=georaster:scott,tiger,,GDAL_IMPORT
SUBDATASET_2_DESC=Table:GDAL_IMPORT
```

B Options de création

- **BLOCKXSIZE :** le nombre de colonne de pixel dans un bloc raster.
- **BLOCKYSIZE :** le nombre de ligne de pixel dans un bloc raster.
- **BLOCKBSIZE :** le nombre de bandes dans un bloc raster.
- **SRID :** assigne une identification de projection/système de référence EPSG à un GeoRaster.
- **INTERLEAVE :** mode d'entrelacement de bande, BAND, LINE, PIXEL (ou BSQ, BIP, BIL) pour les entrelacements de bandes séquentielles, Ligne ou Pixel.
- **DESCRIPTION :** une description simple d'une nouvelle table dans la syntaxe

SQL. Si la table existe déjà, cette option de création sera ignorée, par exemple :

```
% gdal_translate -of georaster landsat_823.tif
geor:scott/tiger@orcl,landsat,raster \
-co DESCRIPTION="(ID NUMBER, NAME VARCHAR2(40), RASTER
MDSYS.SDO_GEORASTER)" \
-co INSERT="VALUES (1,'Scene 823',SDO_GEOR.INIT())"
```

- **INSERT :** une clause SQL simple d'insert/values pour informer le pilote des valeurs qu'il doit insérer dans une nouvelle ligne de la table, par exemple :

```
% gdal_translate -of georaster landsat_825.tif
geor:scott/tiger@orcl,landsat,raster \
-co INSERT="ID, RASTER VALUES (2,SDO_GEOR.INIT())"
```

- **COMPRESS :** options de compression, JPEG-F, JPEG-B, DEFLATE ou NONE. Les deux options JPEG sont avec pertes, ce qui signifie que les pixels originels sont modifiés. Le type JPEG-F stocke une structure complète du JPEG sur chaque bloc tandis que le type JPEG-B est plus petit puisqu'il ne stocke pas les tables de quantification et d'Huffman.
- **QUALITY :** option de la qualité de la compression pour le format JPEG de 0 à 100. 75 par défaut.
- **NBITS :** type de données sous byte, options : 1, 2 ou 4.

C Importer des GeoRaster

Pendant le processus d'import de raster dans un objet GeoRaster il est possible de donner au pilote une simple définition de table SQL et également une clause de valeurs/insert SQL pour informer le pilote de la table à créer et les valeurs à ajouter aux nouvelles lignes. L'exemple suivant réalise cela :

```
% gdal_translate -of georaster landsat_1.tif
georaster:scott/tiger,,landsat,scene \
-co "DESCRIPTION=(ID NUMBER, SITE VARCHAR2(45), SCENE
MDSYS.SDO_GEORASTER)" \
-co "INSERT=VALUES(1,'West fields', SDO_GEOR.INIT())"
```

Notez que l'option de création *DESCRIPTION* nécessite de donner le nom de la table (landsat). Le nom de la colonne (scene) doit correspondre la description :

```
% gdal_translate -of georaster landsat_1.tif
georaster:scott/tiger,,landsat,scene \
-co "DESCRIPTION=(ID NUMBER, SITE VARCHAR2(45), SCENE
MDSYS.SDO_GEORASTER)" \
-co "INSERT=VALUES(1,'West fields', SDO_GEOR.INIT())"
```

Si la table *landsat* existe, l'option *DESCRIPTION* est ignorée. Le pilote peut seulement mette à jour une colonne GeoRaster par commande `gdal_translate`. Oracle créé des noms et valeurs par défauts pour RDT et RID pendant l'initialisation des objet *SDO_GEORASTER* mais les utilisateurs peuvent aussi définir un nom et une valeur de leur choix.

```
% gdal_translate -of georaster landsat_1.tif
```

```
georaster:scott/tiger,,landsat,scene \
-co "INSERT=VALUES(10,'Main building', SDO_GEOR.INIT("RDT", 10))"
```

Si aucune information n'est données sur l'endroit où stocker le raster, le pilote créera (s'il n'existe pas déjà) une table par défaut nommée *GDAL_IMPORT* avec juste une colonne GeoRaster nommée *RASTER*, par exemple :

```
% gdal_translate -of georaster input.tif "geor:scott/tiger@dbdemo"
```

D Exporter GeoRaster

Un GeoRaster peut être identifié par une clause Where ou par une pair de RDT & RID :

```
% gdal_translate -of gtiff geor:scott/tiger@dbdemo,landsat,scene,id=54
output.tif
% gdal_translate -of gtiff geor:scott/tiger@dbdemo,st_rdt_1,130
output.tif
```

E Utilisation générale de GeoRaster

Les GeoRaster peuvent être utilisé dans n'importe quel ligne de commande GDAL avec toutes les options disponibles. Comme une extraction d'une reprojection d'un sous jeu de données d'image :

```
% gdal_translate -of gtiff geor:scott/tiger@dbdemo,landsat,scene,id=54
output.tif \
-srcwin 0 0 800 600

% gdalwarp -of png geor:scott/tiger@dbdemo,st_rdt_1,130 output.png
-t_srs EPSG:9000913
```

Deux GeoRaster différents peuvent être utilisé comme entré et sortie lors de la même opération :

```
% gdal_translate -of georaster
geor:scott/tiger@dbdemo,landsat,scene,id=54
geor:scott/tiger@proj1,projview,image -co \
INSERT="VALUES (102, SDO_GEOR.INIT())"
```

Les applications qui utilisent GDAL peuvent théoriquement lire et écrire du GeoRaster comme tout autre format mais la plupart d'entre eux sont plus enclins à tenter d'accéder aux fichiers sur le système de fichier donc une alternative est de créer un VRT pour représenter la description du GeoRaster, par exemple :

```
% gdal_translate -of VRT geor:scott/tiger@dbdemo,landsat,scene,id=54
view_54.vrt
% openenv view_54.vrt
```

Chapitre XXVI GIF -- Graphics Interchange Format

GDAL gère la lecture et l'écriture des fichiers GIF normaux et interlacés. Les fichiers GIF apparaissent toujours comme ayant une bande de 8 bites de carte de couleur. Les fichiers GIF ne gèrent pas le géoréférencement.

Une image GIF avec une transparence aura cette entrée noté comme ayant une valeur alpha de 0.0 (transparent). Aussi, la valeur transparente sera renvoyé comme la valeur noData pour la bande.

Si un fichier world ESRI existe avec l'extension .gfw, .gifw ou .WLD, il sera lu et utilisé pour établir la géotransformation pour l'image.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraite à partir du fichier, et sera stocké comme XML brute dans le domaine de médatonnées xml:XMP.

A Problème lors de la création

Les fichiers GIF peuvent seulement être créé comme une bande de 8 bit en utilisant le mécanisme « CreateCopy ». S'il est écrit à partir d'un fichier qui n'a pas de carte de couleur, une carte de couleur par défaut en nuance de gris est générée. Les fichiers GIF transparent ne sont pas gérés pour l'instant en création.

- **WORLDFILE=ON :** force la génération d'un fichier World d'ESRI associé (.wld).

Des fichiers interlacés (progressive) peuvent être générés en fournissant l'option *INTERLACING=ON* lors de la création.

À partir de GDAL 1.7.0, la gestion du GIF interne de GDAL a été implémenté en se basant sur les source de la bibliothèque giflib 4.1.6 (écrite par Gershon Elbor, Eric Raymond et Toshio Kuratomi), générant donc des GIG compressé en LZW.

Ce pilote a été écrit grâce au financement de DM Solutions Group (http://www.dmsolutions.ca), et CIET International (http://www.ciet.org).

Lisez également :

- Page principale de giflib

Chapitre XXVII Le format GRASS

GDAL gère d'une manière optionnelle la lecture de couches raster au format GRASS existantes (cells et groupes d'images), mais pas l'écriture et l'export. La gestion des couches rasters au format GRASS est déterminée lorsque la bibliothèque est configurée, et nécessite que libgrass soit pré-installée (lisez les remarques ci-dessous).

A Sélection des rasters de GRASS

Les rasters de GRASS peuvent être sélectionnés de différentes manières.

1 Chemin complet vers le fichier cellhd

Le chemin complet vers le fichier cellhd pour la cellule peut être définie. Ce n'est pas un chemin relatif, ou du moins il doit contenir toutes les parties du chemin dans la base de données avec la racine de la base de données incluse. L'exemple suivant ouvre la cellule « proj_tm » dans le jeu de carte « PERMANENT » de la région « proj_tm » dans la base de données GRASS placée dans */u/data/grassdb*.

Par exemple :

```
% gdalinfo /u/data/grassdb/proj_tm/PERMANENT/cellhd/proj_tm
```

2 Chemin complet vers le répertoire

Le chemin complet vers le répertoire contenant les informations sur le groupe d'image (ou le fichier REF à l'intérieure) peut être définie pour se référer à l'ensemble du groupe comme un simple jeu de données. Les exemples suivants font la même chose.

Par exemple :

```
% gdalinfo /usr2/data/grassdb/imagery/raw/group/testmff/REF
% gdalinfo /usr2/data/grassdb/imagery/raw/group/testmff
```

3 Fichier de configuration

S'il y a un fichier de configuration .grassrc5 correcte dans le répertoire home de l'utilisateur, alors les cellules et les groupes d'images peuvent être ouverte juste avec le nom de la cellule. Cela fonctionne seulement pour les cellules ou les images dans la région en court et le jeu de données comme définie dans le fichier .grassrc5.

B Fonctionnalités gérées

Les fonctionnalités suivantes sont gérées par le lien GDAL/GRASS :

- Jusqu'à 256 entrée dans la carte de couleur des cellules sont lues (0-255).
- Les cellules à entier compressé et non compressé, point flottant et double précision sont toutes supportées. Les cellules d'entier sont classées avec une bande de type « Byte » si le format 1-byte est utilisé. Autrement les cellules d'entiers sont traités comme Uint32 (représente un entier 32-bit non signé).
- Les informations de géo-référencement sont proprement lu à partir de GRASS.
- Un essaie est réalisé pour traduire les systèmes de coordonnées, mais certaines conversions peuvent échouées, en particulier la prise en charge des datums et des unités.

Voyez également :

- Page officielle de GRASS GIS
- page libgrass

Remarques sur les variations des pilotes :

Pour GRASS 5.7, Radim Blazek a déplacé le pilote pour utiliser la bibliothèque partagée de GRASS directement au lieu d'utiliser libgrass. Pour l'instant (GDAL 1.2.2 et suivant) les deux versions du pilote sont disponibles et peut être configuré en utilisant l'option « --with-libgrass » pour la manière avec libgrass ou « --with-grass=<dir> » pour la nouvelle version de la bibliothèque GRASS 5.7. La version du pilote GRASS 5.7 ne gère pas pour l'instant l'accès au système de coordonnées, bien qu'il soit possible que cela soit corrigé.

Chapitre XXVIII GRIB -- WMO General Regularly-distributed Information sous la forme Binaire

GDAL gère la lecture des données raster des formats GRIB1 et GRIB2, avec un embryon de gestion pour le système de coordonnées, le géoréférencement et les autres méta-données. Le format GRIB est communément utilisé pour la distribution d'information météorologique, et est diffusé par l'Organisation Météorologique Mondiale.

Le pilote GRIB de GDAL est basé sur une version modifiée de l'application degrib qui a été écrite par Arthur Taylor du NOAA NWS NDFD (MDL). L'application degrib (et le pilote GRIB de GDAL) est compilé avec la bibliothèque de décodage de grib g2clib écrit initialement par John Huddleston du NOAA NWS NCEP.

Il y a plusieurs schémas d'encodage pour les données raster au format GRIB. La plupart devrait être géré en incluant l'encodage PNG. Les fichiers GRIB encodés en JPEG2000 sont généralement géré si GDAL est également compilé avec la gestion du JPEG2000 par un des pilotes JPGE2000 de GDAL. La bibliothèque JasPer généralement fournie la meilleure gestion du jpeg2000 pour le pilote GRIB.

Les fichiers GRIB peuvent être représenté dans GDAL comme ayant plusieurs bandes, avec un ensemble de bande représentant une séquence temps. Les bandes GRIB sont représentées en Float64 (virgule flottante à double précision) sans regard de leur valeurs réelles. Les méta-données GRIB sont capturés par bande de méta-données et utilisé pour définir des descriptions de bandes, similaire à ceci :

```
Description = 100000[Pa] ISBL="Isobaric surface"
    GRIB_UNIT=[gpm]
    GRIB_COMMENT=Geopotential height [gpm]
    GRIB_ELEMENT=HGT
    GRIB_SHORT_NAME=100000-ISBL
    GRIB_REF_TIME=  1201100400 sec UTC
    GRIB_VALID_TIME=  1201104000 sec UTC
    GRIB_FORECAST_SECONDS=3600 sec
```

Les fichiers GRIB2 peuvent également inclure un extrait du numéro de modèle de définition du produit (octet 8-9), et les valeurs du modèle de définition du produit (> à 10 octets) sous forme de métadonnées comme ceci :

```
GRIB_PDS_PDTN=0
GRIB_PDS_TEMPLATE_NUMBERS=3 5 2 0 105 0 0 0 1 0 0 0 1 100 0 0 1 134
160 255 0 0 0 0 0
```

La bibliothèque que GDAL utilise pour lire les fichiers GRIB est connu pour ne pas être thread-safen vous devez donc éviter de lire et écrite plusieurs jeux de données GRIB au même moment à partir de thread différents.

Lisez également :

- Décodeur GRIB2 de "degrib" NOAA NWS NDFD
- Bibliothèque de décodage de grib par g2clib NOAA NWS NCEP
- Documents sur le format GRIB WMS

Chapitre XXIX Le format Gtiff

La plupart des formes de fichiers TIFF et GeoTIFF sont supportés par GDAL en lecture, et certaines variétés peuvent être écrite. Lorsque GDAL est compilé avec la version interne de la librairie libtiff, ou avec une version de libtiff >= 4.0, GDAL gère également la lecture et l'écriture de fichiers BigTIFF (évolution du format TIFF pour gérer des fichiers de taille supérieure à 4 GO).

Les types de bandes en Byte, UInt16, Int16, UInt32, Int32, Float32, Float64, CInt16, CInt32, CFloat32 et CFloat64 sont supportées en lecture et en écriture. Les images avec une palette retourneront des informations sur la palette associée à la bande. Les formats de compression listés ci-dessous devrait être également supportés en lecture.

De plus, les fichiers de un bite et certains formulations inhabituelles de fichier GeoTIFF, tels que les fichiers avec un modèle de couleur YCbCr, seront automatiquement transformé sous la forme RVBA (Rouge, Vert, Bleu, Alpha), et traité comme quatre bandes de huit bites.

A Géo-référencement

La plupart des projections devrait être supportées, avec le signalement que, dans le but de traduire un système de projection non commune et de coordonnées Géographique en OGC WKT, il est nécessaire d'avoir les fichiers csv EPSG disponible. Ils doivent être trouvés à l'endroit pointé par la variable d'environnement GEOTIFF_CSV.

Le géo-référencement à partir d'un GéoTIFF est supporté sous la forme d'un « point » et d'une taille de pixel, une matrice de transformation ou une liste de points d'amer.

Si aucune information de géo-référencement n'est disponible dans le fichier TIFF en lui-même, GDAL cherchera un fichier world d'ESRI avec l'extension .tfw, .tiffw ou .wld, ou également comme un fichier .tab de MapInfo (seul les points d'amer sont utilisé, les Coordsys seront ignorés).

GDAL peut lire et écrire le *RPCCoefficientTag* comme décrit dans l'extension proposée RPCs in GeoTIFF. La balise est écrite seulement pour les fichiers créés avec le profile par défaut GDALGeoTIFF. Pour les autres profiles, un fichier .RPB est créé. Dans le modèle de données de GDAL, les coefficients RPC sont stockés dans le domaine de métadonnée RPC. Pour plus de détails, voir la RFC de géoréférencement RPC. Si des fichiers .RPB ou _RPC.TXT sont trouvés, ils seront utilisés pour lire les RPCs, même si la balise

RPCCoefficientTag est définie.

B Masques de transparence interne

(*à partir de GDAL 1.6.0*)

Les fichiers TIFF peuvent contenir des masques de transparence internes. Le pilote GeoTIFF reconnait un répertoire interne comme étant un masque de transparence lorsque le bite FILETYPE_MASK est positionné sur l'attribut TIFFTAG_SUBFILETYPE. Selon la spécification TIFF, de tels masques de transparence internes contiennent des données échantillonnées à 1 bite. Bien que la spécification TIFF autorise des résolutions plus grandes les masques de transparence, le pilote GeoTIFF ne gère que ceux qui ont la même dimension que l'image principale. Des masques de transparence internes sont également gérés.

(*à partir de GDAL 1.6.0*) Lorsque la variable d'environnement *GDAL_TIFF_INTERNAL_MASK* est positionnée à YES, et que le fichier GeoTIFF est ouvert en mode mise à jour, la méthode *CreateMaskBand()* appelée sur un fichier TIFF ou une de ses bandes créera un masque de transparence interne. Sinon, le comportement par défaut des masques de transparence sera utilisé, c'est-à-dire la création d'un fichier .msk, comme indiqué dans la RFC 15

À partir de GDAL 1.8.0,une bande de masque interne d'1-bit sont décompressé. Lors de la relecture, pour réaliser la conversion entre la bande de masque et la bande alpha plus facilement, les bandes de masque sont exposées à l'utilisateur tout en étant promus en 8 bits (i.e. la valeur pour les pixels non masqués est de 255) à moins que l'option de configuration *GDAL_TIFF_INTERNAL_MASK_TO_8BIT* est définie à NO. Cela n'affecte pas la manière dont la bande de masque est écrite (c'est toujours 1-bit).

C Aperçus

Le pilote GeoTIFF gère la lecture, la création et la mise à jour d'aperçus internes. Ceux-ci peuvent être créés sur des fichiers GeoTIFF ouverts en mode mise à jour (avec gdaladdo par exemple). Si le fichier GeoTIFF est ouvert en lecture seule, la création d'aperçus sera faite dans un fichier externe .ovr. Les aperçus sont mise à jour uniquement sur requête par appel à la méthode *BuildOverviews()*. Si un fichier GeoTIFF possède un masque de transparence et que la variable d'environnement GDAL_TIFF_INTERNAL_MASK est positionnée à YES et que le fichier est ouvert en mode mise à jour, *BuildOverviews()** créera automatiquement des aperçus pour le masque de transparence interne. Ces aperçus seront rafraichis par des appels ultérieurs à *BuildOverviews()*, même si *GDAL_TIFF_INTERNAL_MASK* n'est pas positionnée à YES.

(À partir de GDAL 1.8.0) La taille du bloc (hauteur et largeur de la tuile) utilisée pour les aperçues (interne ou externe) peut être définie en définissant la variable d'environnement*GDAL_TIFF_OVR_BLOCKSIZE* à une puissance de deux entre 64 et 4096. 128 est la valeur par défaut.

D Métadonnée

GDAL peut faire face aux balises baseline du TIFF comme métadonnées au niveau du jeu de données :

- TIFFTAG_DOCUMENTNAME
- TIFFTAG_IMAGEDESCRIPTION

- TIFFTAG_SOFTWARE
- TIFFTAG_DATETIME
- TIFFTAG_ARTIST
- TIFFTAG_HOSTCOMPUTER
- TIFFTAG_COPYRIGHT
- TIFFTAG_XRESOLUTION
- TIFFTAG_YRESOLUTION
- TIFFTAG_RESOLUTIONUNIT
- TIFFTAG_MINSAMPLEVALUE (lecture seule)
- TIFFTAG_MAXSAMPLEVALUE (lecture seule)

Le nom de l'item de métadonnées est l'un des noms ci-dessus ("TIFFTAG_DOCUMENTNAME", ...).

Les autres items de métadonnées non standard peuvent être stockés dans un fichier TIFF créé avec le profile GDALGeoTIFF (par défaut, voir plus bas dans la section :ref:`gdal.gdal.formats.gtiff.issues`). Ces items de métadonnées sont groupés ensemble dans une chaîne XML stockés dans la balise ASCII non standard *TIFFTAG_GDAL_METADATA/* (code 42112). Quand le profile BASELINE ou GeoTIFF sont utilisé, ces items de métadonnées non standard sont stockés dans un fichier PAM .aux.xml.

Unknown interpreted text role "ref".

La valeur de l'item de métadonnées *GDALMD_AREA_OR_POINT* ("AREA_OR_POINT") est stockée dans la clé GeoTIFF *RasterPixelIsPoint* pour les profiles *GDALGeoTIFF* ou *GeoTIFF*.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraites à partir du fichier et seront stockées dans un contenu brute XML dans le domaine de métadonnées xml:XMP.

E Valeur nodata

GDAL stocke la valeur nodata de la bande dans la balise ASCII non standard *TIFFTAG_GDAL_NODATA* (code 42113) pour les fichiers créés avec le profile par défaut *GDALGeoTIFF*. Notez que toutes les bandes doivent avoir la même valeur nodata. Quand le profile BASELINE ou GeoTIFF sont utilisé, la valeur nodata est stockée dans le fichier PAM .aux.xml file.

F Problèmes de création

Les fichiers GeoTIFF peuvent être créés avec n'importe quel type de bande définie dans GDAL, les types complexes inclus. Les fichiers créés peuvent avoir n'importe quel nombre de bandes. Les fichiers avec exactement trois bandes donneront une interprétation photométrique de RVB, les fichiers avec exactement quatre bandes donneront une interprétation photométrique de RVBA, tandis que toutes les autres combinaisons donneront une interprétation photométrique de MIN_IS_WHITE. Les fichiers avec des tables de pseudo-couleur, ou des points d'amer peuvent, pour l'instant, seulement être créés lors d'une création à partir d'un ensemble de données GDAL avec ces objets (*GDALDriver:CreateCopy()*).

Note!

Notez que le format GeoTIFF ne gère pas la description paramétrique des datums, donc les paramètres TOWGS84 dans

les systèmes de coordonnées sont perdu dans le format GeoTIFF.

1 Options de création

- **TFW=YES :** Force la génération d'un fichier associé world d'ESRI (.tfw). Lisez la section les fichiers world pour plus de détails.
- **INTERLEAVE=[BAND,PIXEL] :** Par défaut les fichiers TIFF avec des pixels entrelacées (PLANARCONFIG_CONTIG dans la terminologie TIFF) sont créés. Ceux-ci sont sensiblement moins efficace que les bandes séparées pour certaines choses, mais certaines applications supporte seulement les fichiers TIFF avec des pixels entrelacés.
- **TILED=YES :** Par défaut des fichiers TIFF « strip » sont créés (NdT : des fichiers « nus », sans tuilage par exemple). Cette option peut être utilisé pour forcer la création de fichiers TIFF tuilés.
- **BLOCKXSIZE=n :** définit la largeur de la tuile, par défaut à 256.
- **BLOCKYSIZE=n :** définit la hauteur de la tuile ou du « strip » [Set tile or strip height]. La hauteur de la tuile est de 256 par défaut, la hauteur du « strip » est par défaut à une valeur de 8K ou inférieure.
- **NBITS=n :** Crée un fichier avec moins de 8 bites par échantillon, en passant une valeur de 1 à 7. Le type du pixel doit être l'octet (Byte). À partir de 1.6.0, les valeurs de n=9...15 (type UInt16) et n=17...31 (type UInt32) sont également accepté.
- **COMPRESS=[JPEG/LZW/PACKBITS/DEFLATE/CCITTRLE/CCITTFAX3/CCITTFAX4/NONE] :** définit la compression à utiliser. JPEG doit seulement être utilisé avec des données en octet (8 bit par canal). Mais à partir de GDAL 1.7.0 et en supposant que GDAL a été compilé avec les bibliothèques internes libtiff et libjpeg, il est possible de lire et écrite les fichiers TIFF avec des fichiers TIFF compressé en JPEG 12 bit (vue comme des bandes UInt16 avec NBITS=12). Voir la page wiki "8 et 12 bit JPEG dans les TIFF" pour plus de détails. La compression CCITT doit être uniquement utilisée avec des données à 1 bite (NBITS=1). La valeur par défaut est aucune compression (NONE).
- **PREDICTOR=[1/2/3] :** définit la [predictory] pour la compression LZW ou DEFLATE. La valeur par défaut est de 1 (pas de prédiction), 2 est la prédiction par différence horizontale et 3 par point flottant. [Set the predictor for LZW or DEFLATE compression. The default is 1 (no predictor), 2 is horizontal differencing and 3 is floating point prediction.]
- **SPARSE_OK=TRUE/FALSE :** (à partir de GDAL 1.6.0) est ce que les fichiers nouvellement créés doivent ils être autorisés à être *sparsé* ? Les fichiers *sparsés* ont 0 tuiles/strip de distance pour les blocs jamais écrit et sauver de l'espace ; cependant, la plupart des paquets hors GDAL ne peuvent pas lire de tels fichiers. *FALSE* par défaut.
- **JPEG_QUALITY=[1-100] :** définit la qualité JPEG lors de l'utilisation de la compression JPEG. Une valeur de 100 est la meilleur qualité (faible compression) et 1 est la moins bonne qualité (meilleure compression). Par défaut la valeur est à 75.
- **ZLEVEL=[1-9] :** définit le niveau de compression à utiliser avec la compression DEFLATE. Une valeur de 9 correspond à la compression la plus forte, 1 à la plus faible. La valeur par défaut est de 6.
- **PROFILE=[GDALGeoTIFF/GeoTIFF/BASELINE] :** contrôle quelles balises inhabituelles sont émises par GDAL.
    - Avec GDALGeoTIFF (la valeur par défaut) plusieurs balises GDAL personnalisées peuvent être écrites.

    - Avec GeoTIFF seulement des balises GeoTIFF seront ajoutés à celle habituelles.
    - Avec BASELINE aucune balises GDAL ou GeoTIFF sera écrites. BASELINE est parfois utile lors de l'écriture de fichier qui seront lu par des applications intolérante aux balises non reconnues.
- **PHOTOMETRIC=[MINISBLACK/MINISWHITE/RGB/CMYK/YCBCR/CIELAB/ICCLAB/ITULAB] :** définit la balise d'interprétation photométrique. Par défaut la valeur est à MINISBLACK, mais si l'image en entrée possède trois ou quatre bandes de type Octet, alors RGB sera utilisé. Vous pouvez écraser la valeur par défaut en utilisant cette option.
- **ALPHA=YES :** Le premier "extrasample" est noté comme étant alpha s'il existe un extra samples. Cela est nécessaire si vous désirez produire un fichier TIFF en nuance de gris avec une bande alpha (par exemple).
- **BIGTIFF=YES/NO/IF_NEEDED/IF_SAFER :** Contrôle si le fichier créé est un fichier BigTIFF ou un TIFF classique.
    - *YES* force le format BigTIFF.
    - *NO* force le format TIFF classique.
    - *IF_NEEDED* créera seulement BigTIFF si cela est clairement nécessaire (non compressé, et des images plus grande que 4 Go).
    - *IF_SAFER* créera un BigTIFF si le fichier résultant *pourrait* excédé 4 Go.

  BigTIFF est une variante du TIFF qui peut contenir plus de 4 Go de données (la taille des TIFF classique est limité à cette valeur). L'option est disponible si GDAL a été complié avec la bibliothèque libtiff 4.0 ou supérieure (ce qui est le cas de la version interne de libtiff à partir de GDAL >= 1.5.0). *IF_NEEDED* par défaut (*IF_NEEDED* et *IF_SAFER* sont disponible à partir de GDAL 1.6.0)
  Lors de la création d'un nouveau GeoTIFF avec aucune compression, GDAL calcul en avance la taille du fichier résultant. Si la taille calculée de ce fichier est supérieur à 4 Go, GDAL décidera automatiquement un fichier BigTIFF. Cependant, quand la compression est utilisée, il n'est pas possible de connaitre à l'avance la taille du fichier, un fichier classique sera alors choisit. Dans ce cas, l'utilisateur doit explicitement choisir la création d'un bigTIFF avec l'option BIGTIFF=YES s'il a anticipé la taille finale du fichier. Si l'option BigTIFF n'a pas été explicitement demandée ou supposée et que le fichier résultant est trop gros pour le fichier classique TIFF, libtiff échouera avec un message d'erreur comme "TIFFAppendToStrip:Maximum TIFF file size exceeded".
- **PIXELTYPE=[DEFAULT/SIGNEDBYTE] :** en définissant ce paramètre à SIGNEDBYTE, un nouveau fichier d'octet peut être écrit en force comme octet signé.
- **COPY_SRC_OVERVIEWS=[YES/NO] :** (GDAL >= 1.8.0, CreateCopy() seulement) en définissant ce paramètre à YES (NO par défaut), les aperçues potentiellement existantes du jeu de données source seront copiées vers le jeu de données cible sans retraitement. Si les aperçues de la bande de masque existe aussi, en supposant que l'option de configuration*GDAL_TIFF_INTERNAL_MASK* est définie à YES, elles seront aussi copiées. Notez que cette option de création n'aura aucun effet si les options générales (i.e. options qui ne sont pas des options de création) de gdal_translate sont utilisées.

G À propos de la compression d'images RVB au format JPEG

Lorsqu'on convertit une image RVB dans le format JPEG-dans-TIFF, utilisez PHOTOMETRIC=YCBCR peut rendre le fichier résultant typiquement de 2 à 3 fois plus

petits que la valeur photométrique par défaut (RGB). Quand on utilise PHOTOMETRIC=YCBCR, l'option INTERLEAVE doit être laissée à sa valeur par défaut (PIXEL), sinon libtiff échouera lors de la compression des données. Prenez note également que les dimensions des tuiles ou des "strips" doivent être un multiple de 8 pour PHOTOMETRIC=RGB ou 16 pour PHOTOMETRIC=YCBCR

1 Options de configuration

Ce paragraphe liste les options de configuration qui peuvent être définie pour modifier le comportement par défaut du pilote GTiff.

Explicit markup ends without a blank line; unexpected unindent.

- **GTIFF_IGNORE_READ_ERRORS :** (GDAL >= 1.9.0) peut être définie à TRUE pour éviter de renvoyer les erreurs libtiff vers les erreurs GDAL. Can help reading partially corrupted TIFF files
- **ESRI_XML_PAM :** peut être définie à TRUE pour forcer l'écriture des métadonnées vers le PAM dans le domaine xml:ESRI.
- **JPEG_QUALITY_OVERVIEW :** entier entre 0 et 100. Valeur par défaut : 75. Qualité des aperçues compressées en JPEG, soit en interne soit en externe.
- **GDAL_TIFF_INTERNAL_MASK :** Voir la section :ref:`gdal.gdal.formats.gtiff.internal_mask`. Valeur par défaut : FALSE.
  Unknown interpreted text role "ref".
- **GDAL_TIFF_INTERNAL_MASK_TO_8BIT :** Voir la section :ref:`gdal.gdal.formats.gtiff.internal_mask`. Valeur par défaut : TRUE
  Unknown interpreted text role "ref".
- **USE_RRD :** peut être définie à TRUE pour forcer les aperçues externes dans le format RRD. Valeur par défaut : FALSE
- **TIFF_USE_OVR :** peut être définie à TRUE pour forcer les aperçues externes dans le format GeoTIFF (.ovr). Valeur par défaut : FALSE
- **GTIFF_POINT_GEO_IGNORE :** peut être définie à TRUE pour revenir au comportement de GDAL < 1.8.0 pour la manière dont les pixels sont interprétés par rapport à la géotransformation. Voir RFC 33: GTiff - Fixing PixelIsPoint Interpretation pour plus de détails. Valeur par défaut : FALSE.
- **GTIFF_REPORT_COMPD_CS :** (GDAL >= 1.9.0). peut être définie à TRUE pour éviter de modifier la verticale du CS dans un composant CS. Valeur par défaut : FALSE
- **GDAL_ENABLE_TIFF_SPLIT :** peut être définie à FALSE pour éviter que des fichiers d'une seule bande soient présentée comme en ayant plusieurs. Valeur par défaut : TRUE

Bullet list ends without a blank line; unexpected unindent.

Explicit markup ends without a blank line; unexpected unindent.

- **GDAL_TIFF_OVR_BLOCKSIZE :** Voir la section :ref:`gdal.gdal.formats.gtiff.apercues`.
  Unknown interpreted text role "ref".
- **GTIFF_LINEAR_UNITS :** peut être définie en BROKEN pour lire les fichiers

GeoTIFF qui ont un easting/northing improprement définie en mètre lorsqu'ils doivent être en unité linéaire du système de coordonnées. (Ticket #3901).

Lisez également :

- Page d'information sur GeoTIFF : http://www.remotesensing.org/geotiff/geotiff.html
- Page libtiff : http://www.remotesensing.org/geotiff/geotiff.html
- Détails du format de fichier BigTIFF : http://www.awaresystems.be/imaging/tiff/bigtiff.html

Chapitre XXX HDF4 --- Hierarchical Data Format Version 4 (HDF4)

Il y a deux formats HDF (4.x et les versions antérieures) et HFD5. Ces formats sont complètement différents et incompatible. Ce pilote a été destiné uniquement pour l'importation des formats de fichier HDF4. Le Système d'Observation de la Terre (EOS, Earth Observing System) de la NASA maintient ses propres modifications HDF appelé HDF-EOS. Ces modifications convient pour les données de capteur distant et est pleinement compatible avec la base du HDF. Ce pilote peut importer des fichiers HDF4-EOS. Pour l'instant EOS utilise HDF4-EOS pour le stockage des données (satellites télémétrique de type 'Terra' et 'Aqua'). Dans le futur ils passeront au format HDF5-EOS, qui sera utilisé pour le satellite télémétrique de type 'Aura'.

A Gestion des Images Multiple (Sous-ensemble de données)

Le Format de Données Hiérarchique (HDF) est un conteneur pour différents sous-ensemble de données. Pour les données stockant des sous-ensemble de Données Scientifique (SDS, Scientific Datasets) est utilisé le plus souvent. SDS est un tableau multi-dimensionnel remplit de données. Un fichier HDF peut contenir différents tableaux SDS. Ceux-ci peuvent se différencier par la taille, le nombre de dimensions et peuvent représenter des données de région différentes.

Si le fichier contient seulement un SDS qui apparaît être une image, il peut accessible normalement, mais s'il contient des images multiples, il peut être nécessaire d'importer le fichier via une procédure en deux étapes. La première est d'obtenir un rapport des composantes images (tableaux SDS) du fichier avec `gdalinfo`, puis d'importer les images désirées avec `gdal_translate`. La commande `gdalinfo` liste tous les sous-ensembles de données multi-dimensionnelles à partir du fichier HDF en entrée. Le nom des images individuelles (sous-ensemble de données) sont assignés à l'objet méta-données **SUBDATASET_n_NAME**. La description de chaque image est trouvé dans l'objet méta-données **SUBDATASET_n_DESC**. Pour les images HDF4, le nom du sous-ensemble de données sera formaté comme suit :

```
//HDF4_SDS:subdataset_type:file_name:subdataset_index//
```

où `subdataset_type` montre des noms prédéfinis pour quelques sous-ensemble de

données HDF bien connus, `file_name` est le nom du fichier en entrée et `subdataset_index` est l'index de l'image à utiliser (pour utilisation interne dans GDAL).

Pour la seconde étape, vous devez fournir ce nom à `gdalinfo` ou `gdal_translate` pour la lecture des données. Par exemple, nous voulons lire des données à partir du sous-ensemble de données MODIS Level 1B :

```
$ gdalinfo GSUB1.A2001124.0855.003.200219309451.hdf
Driver: HDF4/Hierarchical Data Format Release 4
Size is 512, 512
Coordinate System is `'
Metadata:
    HDFEOSVersion=HDFEOS_V2.7
    Number of Scans=204
    Number of Day mode scans=204
    Number of Night mode scans=0
    Incomplete Scans=0
...beaucoup de méta-données sautées ...
Subdatasets:
    SUBDATASET_1_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:0
    SUBDATASET_1_DESC=[408x271] Latitude (32-bit floating-point)
    SUBDATASET_2_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:1
    SUBDATASET_2_DESC=[408x271] Longitude (32-bit floating-point)
    SUBDATASET_3_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:2
    SUBDATASET_3_DESC=[12x2040x1354] EV_1KM_RefSB (16-bit unsigned
integer)
    SUBDATASET_4_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:3
    SUBDATASET_4_DESC=[12x2040x1354] EV_1KM_RefSB_Uncert_Indexes (8-
bit unsigned integer)
    SUBDATASET_5_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:4
    SUBDATASET_5_DESC=[408x271] Height (16-bit integer)
    SUBDATASET_6_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:5
    SUBDATASET_6_DESC=[408x271] SensorZenith (16-bit integer)
    SUBDATASET_7_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:6
    SUBDATASET_7_DESC=[408x271] SensorAzimuth (16-bit integer)
    SUBDATASET_8_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:7
    SUBDATASET_8_DESC=[408x271] Range (16-bit unsigned integer)
    SUBDATASET_9_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.20021
9309451.hdf:8
    SUBDATASET_9_DESC=[408x271] SolarZenith (16-bit integer)
    SUBDATASET_10_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:9
    SUBDATASET_10_DESC=[408x271] SolarAzimuth (16-bit integer)
    SUBDATASET_11_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:10
    SUBDATASET_11_DESC=[408x271] gflags (8-bit unsigned integer)
    SUBDATASET_12_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:12
    SUBDATASET_12_DESC=[16x10] Noise in Thermal Detectors (8-bit
```

```
unsigned integer)
    SUBDATASET_13_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:13
    SUBDATASET_13_DESC=[16x10] Change in relative responses of thermal
detectors (8-bit unsigned integer)
    SUBDATASET_14_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:14
    SUBDATASET_14_DESC=[204x16x10] DC Restore Change for Thermal Bands
(8-bit integer)
    SUBDATASET_15_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:15
    SUBDATASET_15_DESC=[204x2x40] DC Restore Change for Reflective
250m Bands (8-bit integer)
    SUBDATASET_16_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:16
    SUBDATASET_16_DESC=[204x5x20] DC Restore Change for Reflective
500m Bands (8-bit integer)
    SUBDATASET_17_NAME=HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.2002
19309451.hdf:17
    SUBDATASET_17_DESC=[204x15x10] DC Restore Change for Reflective
1km Bands (8-bit integer)
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0,  512.0)
Upper Right (  512.0,    0.0)
Lower Right (  512.0,  512.0)
Center      (  256.0,  256.0)
```

Maintenant sélectionnons un sous-ensemble de données, décrit comme [12x2040x1354] EV_1KM_RefSB (16-bit unsigned integer) :

```
$ gdalinfo
HDF4_SDS:MODIS_L1B:GSUB1.A2001124.0855.003.200219309451.hdf:2
Driver: HDF4Image/HDF4 Internal Dataset
Size is 1354, 2040
Coordinate System is `'
Metadata:
    long_name=Earth View 1KM Reflective Solar Bands Scaled Integers
...méta-données sautées ...
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0, 2040.0)
Upper Right ( 1354.0,    0.0)
Lower Right ( 1354.0, 2040.0)
Center      (  677.0, 1020.0)
Band 1 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 2 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 3 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 4 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 5 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 6 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 7 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 8 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 9 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 10 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
Band 11 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
```

```
Band 12 Block=1354x2040 Type=UInt16, ColorInterp=Undefined
```

Ou vous pouvez utiliser `gdal_translate` pour lire les bandes de l'image à partir de ce sous-ensemble de données. Notez que vous devez fournir exactement le contenu de la ligne marqué **SUBDATASET_n_NAME** à GDAL, en incluant le préfixe `HDF4_SDS:`.

Ce pilote a seulement pour but d'importer des sous-ensembles de capteur distant et géospatiale sous forme d'image raster. Si vous voulez explorer toutes les données contenu dans un fichier HDF vous devez utiliser un autre outil (vous pouvez trouver des informations sur différents outils HDF en utilisant les liens à la fin de cette section).

B Géo-référencement

Il n'y a pas de manière universelle pour stocker le géo-référencement dans les fichiers HDF. Cependant, certains types de produits ont des mécanismes pour sauvegarder le géo-référencement, et certains sont gérés par GDAL. Pour l'instant, sont supportés (subdataset_type est indiqué entre parenthèses) :

- fichiers HDF4 créés par GDAL (**GDAL_HDF4**)
- ASTER Level 1A (**ASTER_L1A**)
- ASTER Level 1B (**ASTER_L1B**)
- ASTER Level 2 (**ASTER_L2**)
- ASTER DEM (**AST14DEM**)
- MODIS Level 1B Earth View products (**MODIS_L1B**)
- MODIS Level 3 products (**MODIS_L3**)
- SeaWiFS Level 3 Standard Mapped Image Products (**SEAWIFS_L3**)

Par défaut le pilote hdf4 lit seulement les points d'amer à partir de toutes les 10 lignes et colonne à partir du jeu de données EOS_SWATH. Vous pouvez changer ce comportement en définissant la variable d'environnement *GEOL_AS_GCPS* à PARTIAL (défaut), NONE, ou FULL.

C Problèmes de création

Ce pilote support la création des ensembles de données Scientifique HDF4. Vous pouvez créer un ensemble de données 2D (un pour chaque bande en entrée) ou un simple ensemble de données 3D ou la 3ème dimension représente le numéro de la bande. Toutes les méta-données et les descriptions des bandes des ensembles de données en entrée sont stockés comme des attributs HDF4. Les informations de projection (s'ils existent) et les coefficients de transformation affine sont aussi stockés sous forme d'attributs. Les fichiers créés par GDAL ont un attribut spécial :

```
"Signature=Created with GDAL (http://www.remotesensing.org/gdal/)"
```

et sont automatiquement reconnus lors de la lecture, ainsi les informations de projections et la matrice de transformation sont restaurés.

Options de création :

- **RANK=n :** créé un SDS à n-dimension. Pour l'instant seul les ensembles de données 2D et 3D sont gérés. Par défaut un ensemble de données à 3 dimensions sera créée.

D Méta-données

Tous les attributs HDF4 sont traduit en transparence comme des méta-données GDAL. Dans les fichiers HDF, les attributs peuvent être assigné à l'ensemble du fichier autant qu'à des sous-ensemble de données particuliers.

E Compilation du pilote

Ce pilote a été compilé au plus haut de la bibliothèque NCSA HDF, vous avez donc besoin de compiler GDAL avec le support HDF4. Vous pouvez chercher un binaire pré-compilé pour votre distribution ou télécharger le code source ou les binaires de la page de NCSA HDF (voyez les liens ci-dessous).

Noter que la bibliothèque NCS HDF a été compilé avec de nombreux paramètres par défaut dans le fichier *hlimits.h*. Par exemple, *hlimits.h* définie le nombre maximal de fichiers ouverts :

```
#   define MAX_FILE   32
```

Si vous avez besoin d'ouvrir plus de fichier HDF4 simultanément, vous devez changer cette valeur et recompiler la bibliothèque HDF4 (et relier GDAL si vous utilisez des bibliothèques HDF statiques).

F Voir aussi

- Implémenté comme *gdal/frmts/hdf4/hdf4dataset.cpp* et *gdal/frmts/hdf4/hdf4imagedataset.cpp*.
- Group HDF
- Sources de donnée aux formats HDF4 et HDF4-EOS : - Earth Observing System Data Gateway
- Documentation de produits individuels, géré par ce pilote : - Geo-Referencing ASTER L1B Data - ASTER Standard Data Product Specifications Document - MODIS Level 1B Product Information and Status - MODIS Ocean User's Guide

Chapitre XXXI HDF5 --- Hierarchical Data Format Release 5 (HDF5)

Ce pilote a pour objectif d'importer le format de fichiers HDF5. Cette modification est disponible pour l'utilisation de données de capteur distant et complètement compatible avec le format HDF5 sous jascent. Ce pilote peut importer les fichiers HDF5-EOS. Actuellement EOS utilise HDF5 pour le stockage de données (télémétrie à partir des satellites 'Aura'). Dans le future ils passeront au format HDF5 qui sera utilisé pour la télémétrie à partir des satellites 'Aura' .

A Gestion des images multiples (Sous jeux de données)

Hierarchical Data Format est un conteneur pour plusieurs jeux de données différents. Pour le stockage des données. HDF contient des tableaux multidimensionnels remplit par des données. Un fichier HDF peut contenir plusieurs tableaux. Ceux-ci peuvent être différent en taille et dimensions.

La première étape est d'avoir un rapport des images composants (tableau) dans le fichier en utilisant `gdalinfo` puis d'importer les images désirées en utilisant `gdal_translate`. La commande `gdalinfo` liste tous les sous jeux de données multidimensionnels à partir du fichier HDF en entrée. Le nom des images individuelles (sous jeux de données) sont assignés à l'objet de méta-données *SUBDATASET_n_NAME*. La description pour chaque image est trouvée dans l'objet méta-données *SUBDATASET_n_DESC*. Pour les images HDF5 les noms du sous jeu de données sera formaté comme cela :

```
HDF5:file_name:subdataset
```

où :

- **file_name** est le nom du fichier d'entrée ;
- **subdataset** est le nom du jeu de données du tableau à utiliser (pour utilisation interne dans GDAL).

À la deuxième étape vous devez fournir ce nom pour `gdalinfo` ou `gdal_translate` pour une lecture réelle des données.

Par exemple, nous voulons lire des données à partir du jeu de données OMI/Aura Ozone (O3) :

```
$ gdalinfo OMI-Aura_L2-OMTO3_2005m0326t2307-o03709_v002-
2005m0428t201311.he5
Driver: HDF5/Hierarchical Data Format Release 5
Size is 512, 512
Coordinate System is `'

Subdatasets:
    SUBDATASET_1_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/APrioriLayerO3
    SUBDATASET_1_DESC=[1496x60x11]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/APrioriLayerO3 (32-
bit floating-point)
    SUBDATASET_2_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/AlgorithmFlags
    SUBDATASET_2_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/AlgorithmFlags (8-bit
unsigned character)
    SUBDATASET_3_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/CloudFraction
    SUBDATASET_3_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/CloudFraction (32-bit
floating-point)
    SUBDATASET_4_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/CloudTopPressure
    SUBDATASET_4_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/CloudTopPressure (32-
bit floating-point)
    SUBDATASET_5_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/ColumnAmountO3
    SUBDATASET_5_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/ColumnAmountO3 (32-
bit floating-point)
    SUBDATASET_6_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/LayerEfficiency
    SUBDATASET_6_DESC=[1496x60x11]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/LayerEfficiency (32-
bit floating-point)
    SUBDATASET_7_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/NValue
    SUBDATASET_7_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/NValue (32-bit
floating-point)
    SUBDATASET_8_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
```

```
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/O3BelowCloud
    SUBDATASET_8_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/O3BelowCloud (32-bit
floating-point)
    SUBDATASET_9_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/QualityFlags
    SUBDATASET_9_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/QualityFlags (16-bit
unsigned integer)
    SUBDATASET_10_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/Reflectivity331
    SUBDATASET_10_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/Reflectivity331 (32-
bit floating-point)
    SUBDATASET_11_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/Reflectivity360
    SUBDATASET_11_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/Reflectivity360 (32-
bit floating-point)
    SUBDATASET_12_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/Residual
    SUBDATASET_12_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/Residual (32-bit
floating-point)
    SUBDATASET_13_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/ResidualStep1
    SUBDATASET_13_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/ResidualStep1 (32-bit
floating-point)
    SUBDATASET_14_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/ResidualStep2
    SUBDATASET_14_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/ResidualStep2 (32-bit
floating-point)
    SUBDATASET_15_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/SO2index
    SUBDATASET_15_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/SO2index (32-bit
floating-point)
    SUBDATASET_16_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
```

```
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/Sensitivity
    SUBDATASET_16_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/Sensitivity (32-bit
floating-point)
    SUBDATASET_17_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/StepOneO3
    SUBDATASET_17_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/StepOneO3 (32-bit
floating-point)
    SUBDATASET_18_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/StepTwoO3
    SUBDATASET_18_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/StepTwoO3 (32-bit
floating-point)
    SUBDATASET_19_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/TerrainPressure
    SUBDATASET_19_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/TerrainPressure (32-
bit floating-point)
    SUBDATASET_20_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/UVAerosolIndex
    SUBDATASET_20_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/UVAerosolIndex (32-
bit floating-point)
    SUBDATASET_21_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/dN_dR
    SUBDATASET_21_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/dN_dR (32-bit
floating-point)
    SUBDATASET_22_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields
/dN_dT
    SUBDATASET_22_DESC=[1496x60x12]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Data_Fields/dN_dT (32-bit
floating-point)
    SUBDATASET_23_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/GroundPixelQualityFlags
    SUBDATASET_23_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/GroundPixelQua
lityFlags (16-bit unsigned integer)
    SUBDATASET_24_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5":/HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
```

```
_Fields/Latitude
    SUBDATASET_24_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/Latitude (32-
bit floating-point)
    SUBDATASET_25_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/Longitude
    SUBDATASET_25_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/Longitude (32-
bit floating-point)
    SUBDATASET_26_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/RelativeAzimuthAngle
    SUBDATASET_26_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/RelativeAzimut
hAngle (32-bit floating-point)
    SUBDATASET_27_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/SolarAzimuthAngle
    SUBDATASET_27_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/SolarAzimuthAn
gle (32-bit floating-point)
    SUBDATASET_28_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/SolarZenithAngle
    SUBDATASET_28_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/SolarZenithAng
le (32-bit floating-point)
    SUBDATASET_29_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/TerrainHeight
    SUBDATASET_29_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/TerrainHeight
(16-bit integer)
    SUBDATASET_30_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/ViewingAzimuthAngle
    SUBDATASET_30_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/ViewingAzimuth
Angle (32-bit floating-point)
    SUBDATASET_31_NAME=HDF5:"OMI-Aura_L2-OMTO3_2005m0113t0224-
o02648_v002-
2005m0625t035355.he5"://HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation
_Fields/ViewingZenithAngle
    SUBDATASET_31_DESC=[1496x60]
//HDFEOS/SWATHS/OMI_Column_Amount_O3/Geolocation_Fields/ViewingZenithA
ngle (32-bit floating-point)
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0,  512.0)
Upper Right (  512.0,    0.0)
```

```
Lower Right (  512.0,  512.0)
Center      (  256.0,  256.0)
```

Maintenant sélectionnons un des sous jeu de données, décrit comme [1645x60] CloudFraction (32-bit floating-point) :

```
$ gdalinfo HDF5:"OMI-Aura_L2-OMTO3_2005m0326t2307-o03709_v002-
2005m0428t201311.he5":CloudFraction
Driver: HDF5Image/HDF5 Dataset
Size is 60, 1645
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        TOWGS84[0,0,0,0,0,0,0],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9108"]],
    AXIS["Lat",NORTH],
    AXIS["Long",EAST],
    AUTHORITY["EPSG","4326"]]
GCP Projection = GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],TOWGS84[0,0,0,0,0,
0,0],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","
8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9108"]],AXI
S["Lat",NORTH],AXIS["Long",EAST],AUTHORITY["EPSG","4326"]]
GCP[  0]: Id=, Info=
        (0.5,0.5) -> (261.575,-84.3495,0)
GCP[  1]: Id=, Info=
        (2.5,0.5) -> (240.826,-85.9928,0)
GCP[  2]: Id=, Info=
        (4.5,0.5) -> (216.754,-86.5932,0)
GCP[  3]: Id=, Info=
        (6.5,0.5) -> (195.5,-86.5541,0)
GCP[  4]: Id=, Info=
        (8.5,0.5) -> (180.265,-86.2009,0)
GCP[  5]: Id=, Info=
        (10.5,0.5) -> (170.011,-85.7315,0)
GCP[  6]: Id=, Info=
        (12.5,0.5) -> (162.987,-85.2337,0)
```

... 3 000 points d'amer sont lu à partir du fichier si le tableau de Latitude et Longitude sont présent :

```
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0, 1645.0)
Upper Right (   60.0,    0.0)
Lower Right (   60.0, 1645.0)
Center      (   30.0,  822.5)
Band 1 Block=60x1 Type=Float32, ColorInterp=Undefined
Open GDAL Datasets:
```

```
  1 N DriverIsNULL 512x512x0
```

vous pouvez utiliser `gdal_translate` pour la lecture des bandes d'images à partir de ce jeu de données.

Notez que vous devez fournir le contenu exact de la ligne noté **SUBDATASET_n_NAME** à GDAL, incluant le préfixe *HDF5:*.

Ce pilote a seulement pour objectif d'importer des jeux de données géospatiaux et de capteur distant sous la forme d'images raster (tableaux 2D ou 3D). Si vous voulez explorer toutes les données dans un fichier HDF vous devez utiliser un autre outil (vous pouvez trouver plus d'informations sur les différents outils HDF en utilisant les liens en bas de cette page).

B Géoréferencement

Il n'y a pas de moyen universel de stockage du géoréférencement dans les fichiers HDF. Cependant, certains types de produits ont des mécanismes pour la sauvegarde du géoréférencement, et certains sont gérés par GDAL. Actuellement ceux gérés sont (subdataset_type affiché entre parenthèses):

- HDF5 OMI/Aura Ozone (O3) Total Column 1-Orbit L2 Swath 13x24km (Level-2 OMTO3)

C Méta-données

Aucune méta-données n'est lues pour l'instant à partir des fichiers HDF5.

D Compilation du pilote

Ce pilote est compilé au dessus de la bibliothèque HDF5 NCSA, vous avez donc besoin de télécharger la bibliothèque HDF5-1.6.4 ou plus récent. Vous avez aussi besoin des bibliothèques zlib 1.2 et szlib 2.0. Pour les utilisateurs Windows assurez vous d'avoir définie les attributs d'écriture (spécialement si vous utilisez cygwin) et que les DLL peuvent être localisé par votre variable d'environnement PATH. Vous pouvez aussi télécharger le code source de la page HDF NCSA (voir lien ci-dessous).

E Voir également

- Implementé dans *gdal/frmts/hdf5/hdf5dataset.cpp and gdal/frmts/hdf5/hdf5imagedataset.cpp*.
- Page de téléchargement de NCSA HDF5 sur le site National Center for Supercomputing Applications
- HDFView est un outil de visualisation pour la navigation et l'édition de fichiers NCSA HDF4 et HDF5.
- Documentation pour les produits individuels, géré par ce pilote : - Aura OMI Total Ozone Data Product-OMTO3

Chapitre XXXII HF2 -- HF2/HFZ heightfield raster

(Disponible à partir de GDAL >= 1.8.0)

GDAL gère la lecture et l'écriture des jeux de données raster heightfield HF2/HFZ/HF2.GZ .

HF2 est un format heightfield qui enregistre les différences entre les valeurs des cellules consécutives. Les fichiers HF2 peuvent aussi être optionnellement compressés par l'algorithme gzip, et donc les fichiers HF2.GZ (ou HFZ, équivalent) peuvent être significativement plus petit que les données non compressées. Le format de fichier permet à l'utilisateur d'avoir un contrôle sur la véracité désirée via le paramètre de précision verticale.

GDAL peut lire et écrire les informations de géoréférencement via les blocs d'en-tête étendue.

A Options de création

- **COMPRESS=YES/NO :** si le fichier doit être compressé avec GZip ou non. NO par défaut.
- **BLOCKSIZE=>block_size :** taille des tuiles internes. Doit être >= 8. 256 par défaut.
- **VERTICAL_PRECISION=vertical_precision :** doit être > 0. 0.01 par défaut.

Augmenter la précision verticale (*[NdT] vertical precision*, i.e. le degré de reproductibilité) diminue la taille du fichier, spécialement avec COMPRESS=YES, mais avec une perte de précision (*[NdT] accuracy*, i.e. la véracité).

B Voir également

- Spécification du format HF2/HFZ
- Spécification des blocs d'en-tête étendue de HF2

Chapitre XXXIII HFA -- Erdas Imagine .img

GDAL gère le format Erdas Imagine .img en lecture et écriture. Le pilote gère la lecture des aperçues, les palettes et le géoréférencement. Il gère les types de bandes erdas u8, s8, u16, u32, f32, f64, c64 et c128.

Les tuiles compressées et manquante dans les fichiers erdas devraient être prises en charge proprement pendant la lecture. Les fichiers entre 2Go et 4 Go devrait fonctionner sous Windows NT et peuvent fonctionner sous certaines plateformes Unix. Les fichiers avec des fichiers étendues externes (nécessaire pour les jeux de données plus grands que 2 Go) sont aussi géré en lecture et écriture.

La lecture et l'écriture des méta-données est gérée au niveau du jeu de données et pour les bandes mais ce sont des méta-données spécifiques à GDAL – pas des méta-données sous une forme reconnus par Imagine. Les méta-données sont stockés dans une table appelée GDAL_MetaData avec une colonne pour une méta-donnée. Le titre est la clé et la valeur de la ligne 1 est la valeur.

A Problème lors de la création

Les fichiers Erdas Imagine peuvent être crées avec n'importe quel type de bande définis par GDAL, les types complexes compris. les fichiers crées peuvent avoir n'importe quel nombre de bandes. les tables pseudo-couleur seront écrites si la méthodologie de *GDALDriver::CreateCopy()* est utilisée. La plupart des projections doivent être gérées bien que la traduction des datums inhabituels (autre que WGS84, WGS72, NAD83 et NAD27) peuvent être problématiques.

Options de création :

- **BLOCKSIZE=blocksize :** Tile width/height (32-2048). Défaut=64
- **USE_SPILL=YES :** force la génération de fichier de débordement (par défaut un fichier de débordement est créé pour des images supérieures à 2 Go seulement). Défaut à *NO*
- **COMPRESSED=YES :** crée un fichier compressé. L'utilisation de fichier de débordement désactive la compression. Défaut à *NO*
- **NBITS=1/2/4 :** crée un fichier avec des types de données avec un sous-bytes spécial.
- **PIXELTYPE=[DEFAULT/SIGNEDBYTE] :** en définissant ce paramètre à

SIGNEDBYTE, un nouveau fichier Byte peut être créé obligatoirement comme byte signé.
- **AUX=YES :** pour créer un fichier .aux. Défaut à *NO*
- **IGNOREUTM=YES :** ignore UTM lors de la sélection du système de coordonnées - utilisera Transverse Mercator. Utilisé seulement pour la méthode *Create()*. Défaut à *NO*.
- **STATISTICS=YES :** pour générer des statistiques et un histogramme. Défaut à *NO*.
- **DEPENDENT_FILE=filename :** nom du fichier dépendant (ne doit pas avoir de chemin absolu), optionnel.
- **FORCETOPESTRING=YES :** force l'utilisation d chaîne ArcGIS PE dans le fichier au lieu de format de système de coordonées Imagine. Dans certains cas cela améliore la compatibilité du système de coordonnées d'ArcGIS.

Erdas Imagine gère la création externe des aperçus (avec `gdaladdo` par exemple). Pour forcer leur création dans un fichier .rrd (plutôt qu'à l'intérieure du fichier .img originel) définissez l'option de configuration globale HFA_USE_RRD=YES.

Les noms des couches peuvent être définis et récupérés avec l'appel de *GDALSetDescription/GDALGetDescription* sur les objets bandes du Raster.

Lisez également :

- Implémenté dans *gdal/frmts/hfa/hfadataset.cpp*.
- Plus d'information, et d'autres outils sont disponibles sur la page du lecteur d'Imagine (img) : http://home.gdal.org/projects/imagine/hfa_index.html
- Site officiel d'Erdas : http://www.erdas.com/

Chapitre XXXIV RST --- Idrisi Raster Format

Ce format est typiquement un format brute. Il y a juste une bande par fichier, sauf dans les types de données RGB24 où les bandes rouges, vertes et bleues sont stockées entrelacées par pixel dans l'ordre Bleu, Vert et Rouge. Les autres types de données sont des entiers non signés sur 8 bits avec des valeurs de 0 à 255 ou des entiers signés sous 16 bits avec des valeurs de -32.768 à 32.767 ou des points à précisions flottantes de 32 bits. La description du fichier est stockée dans un fichier texte d'accompagnement, avec une extension RDC.

Le fichier RDC de description de l'image n'inclut pas de table de couleur, ou d'informations détaillées sur le géoréférencement. La table de couleurs présente peut être obtenu par un autre fichier d'accompagnement en utilisant le même nom que le fichier RST et l'extension SMP.

Pour l'identification des références géographiques, le fichier RDC contient des informations qui renvoient vers un fichier qui détient les détails des références géographiques. Ces fichiers utilisent l'extension REF et résident dans le même répertoire que l'image RST ou plus probablement dans les répertoires d'installation d'Idrisi.

Par conséquent la présence ou l'absence du logiciel Idrisi dans le système d'exploitation déterminera la manière dont ce pilote fonctionnera. En définissant la variable d'environnement IDRISIDIR pour pointer vers le répertoire d'installation principale d'Idrisi cela permettra à GDAL de trouver plus d'informations plus détaillées sur les références géographiques et les projections dans les fichiers REF.

Notez que le pilote RST reconnaît les conventions de nom utilisés dans Idrisi pour les références géographique UTM et Plate carré (State Plane), il n'a donc pas besoin d'accéder aux fichiers REF. C'est le cas pour le fichier RDC qui définie « utm-30n » et « spc87ma1 » dans le champ « ref. system ». Notez que l'export vers le format RST de n'importe quel système de référence géographique générera une suggestion pour le contenu du fichier REF dans la section commentaire du fihcier RDC.

- « .rst » le fichier image brute
- « .rdc » le fichier descriptif
- « .smp » le fichier de la table de couleur
- « .ref » le fichier de référence géographiques

Voir également :

- Implémenté dans *gdal/frmts/idrisi/idrisiraster.cpp*.
- www.idrisi.com

A ILWIS -- Raster Map

Ce pilote implémente la lecture et l'écriture des cartes raster ILWIS et les listes des cartes. Sélectionnez les fichiers raster avec les extensions *.mpr* (pour les cartes raster) ou *.mpl* (pour les listes de cartes).

1 Fonctionnalités :

- Gère les types de données de pixel en octet, Int16, Int32 et Float64.
- Gère les listes de carte avec les ensembles de cartes raster ILWIS associés.
- Lit et écrit le géoréférencement (.grf). Gestion des transformations de géoréférencement est limitée seulement au GeoRefCorner orienté Nord. Si possible, la transformation affine est calculée à partir des coordonnées des coins.
- Lit et écrit les fichiers de coordonnées (.csy). La gestion est limitée au : type de projection de la projection et au type Lat/Lon qui sont définis dans le fichier .csy, le reste des types de projection pré-définie est ignoré.

2 Limitations :

- Les listes de cartes avec le stockage des cartes raster internes (tels que ceux produits par Import General Raster) ne sont pas gérées.
- Les fichiers de domaine ILWIS (.dom) et de représentation (.rpr) sont actuellement ignorés.

Note : Implémenté dans *gdal/frmts/ilwis*.

Voir également :

- http://www.itc.nl/ilwis/default.asp.

Chapitre XXXV INGR --- Intergraph Raster Format

Le format est géré en lecture et écriture.

Le format de fichier Raster d'Intergraph était le format de fichier natif utilisé par les applications logicielles d'Intergraphe pour stocker les données raster. Il se manifeste dans plusieurs formats de données internes.

A Lecture des fichiers INGR

Ceci sont les formats de données que le pilote INGR gère en lecture :

- 2 - Entier en Byte
- 3 - Word Integer
- 4 - 32 bit entiers
- 5 - Point à virgule flottante 32 bit
- 6 - Point à virgule flottante 64 bit
- 9 - Run Length Encoded
- 10 - Run Length Encoded Color
- 24 - CCITT Group 4
- 27 - RVB adaptif
- 28 - 24 bit non compressé
- 29 - Nuance de gris adaptif
- 30 - JPEG GRIS
- 31 - JPEG RVB
- 32 - JPEG CYMK
- 65 - Tuilé
- 67 - Ton continu

Le format "65 - Tuilé" n'est pas un format ; c'est juste une indication que le fichier est tuilé. Dans ce cas l'en-tête des tuiles contient le véritable code de format des données qui peut être n'importe quel code cité au-dessus. Le pilote INGR peut lire les instances tuilés et non tuilés de n'importe quels formats de données gérés.

B Écrire des fichiers INGR

Ceci est la liste des formats que le pilote INGR gère en écriture :

- 2 - Byte Integer
- 3 - Word Integers
- 4 - Integers 32Bit
- 5 - Floating Point 32Bit
- 6 - Floating Point 64Bit

Notez que l'écriture dans ce format n'est pas encouragé !

C Extension de fichier

Ce qui suit est une partie de la liste des extensions de fichier INGR :

.cot	données 8-bit nuance de couleur ou table de couleur
.ctc	8-bit nuance de couleur utilisant une compression PackBits-type (pas commun)
.rgb	24-bit couleur et nuance de gris (non compressé et compression PackBits-type)
.ctb	données 8-bit table de couleur (non compressé ou encodé en run-length)
.grd	données 8, 16 et 32 bit élévation
.crl	données 8 ou 16 bit, run-length compressé nuance de gris ou table de couleur
.tpe	données 8 ou 16 bit, run-length compressé nuance de gris ou table de couleur
.lsr	données 8 ou 16 bit, run-length compressé nuance de gris ou table de couleur
.rle	données 1-bit run-length compressé (16-bit runs)
.cit	données CCITT G3 ou G4 1-bit
.g3	données CCITT G3 1-bit
.g4	données CCITT G4 1-bit
.tg4	données CCITT G4 1-bit (tuilé)
.cmp	JPEG nuance de gris, RGB, ou CMYK
.jpg	JPEG nuance de gris, RGB, ou CMYK

Source : http://www.oreilly.com/www/centers/gff/formats/ingr/index.htm

Le pilote INGR ne nécessite pas une extension de fichier spécifique dans le but d'identifier ou de créer un fichier INGR.

D Géoréférencement

Le pilote INGR ne gère pas la lecture ou l'écriture d'information géoréférencée. La raison est qu'il n'y a pas de manières universelle de stocker le géoréférencement dans les fichiers INGR. Il est possible d'avoir des données géoréférencées stocké dans un fichier .dgn d'accompagnement ou dans le stockage des données spécifiques à l'application dans le fichier lui même.

E Voir également

Pour plus d'information :

- Implémenté dans *gdal/frmts/ingr/intergraphraster.cpp*.
- www.intergraph.com
- http://www.oreilly.com/www/centers/gff/formats/ingr/index.htm
- Fichier des spécifications : ftp://ftp.intergraph.com/pub/bbs/scan/note/rffrgps.zip/

Chapitre XXXVI JAXA PALSAR Processed Products

Ce pilote fournie une gestion améliorée pour les produits PALSAR traités à partir du processeur JAXA PALSAR. Cela contient des produits récupérés des organisations suivantes :

- JAXA (Japanese Aerospace eXploration Agency)
- AADN (Alaska Satellite Facility)
- ESA (European Space Agency)

Ce pilote ne gère pas les produits créer en utilisant le processeur Vexcel (c'est à dire les produits distribués par ERSDAV et les organisations affiliées).

La gestion est fournie pour les fonctionnalités suivantes des produits PALSAR :

- Lecture des produits traités de niveau 1.1 et 1.5 ;
- Géoréférencement pour les produits de niveau 1.5 ;
- Méta-données basique (information des capteurs, espacement des pixels au sol, etc.)
- Données multi-canal (i.e. jeux de données dual-polarisation ou complètement polarimétrique)

C'est un pilote en lecture seul.

Pour ouvrir un produit PALSAR, sélectionnez le répertoire du volume (par exemple, *VOL-ALPSR000000000-P1.5_UA* ou *VOL-ALPSR000000000-P1.1__A*). Le pilote utilisera alors les informations contenu dans le fichier du répertoire du volume pour trouver les fichiers images (les fichiers IMG-*), ansi que les fichiers *Leader*. Notez que le fichier *Leader* est essentiel pour des opérations correctes du pilote.

Lisez également :

- Données échantillon RESTEC

Chapitre XXXVII JP2ECW -- ERMapper JPEG2000 (.jp2)

GDAL gère la lecture et l'écriture des fichiers JPEG2000 en utilisant le SKD ECW d'ERDAS.

Le système de coordonnées et de transformation du géoréférencement sont lu, et un certain degré de gestion est inclus pour GeoJP2 (tm) (GeoTIFF-in-JPEG2000), ERDAS GML-in-JPEG2000, et la nouvelle spécification GML-in-JPEG2000 développé par l'OGC. La gestion du pilote JP2ECW dans GDAL est optionnelle et nécessite la liaison vers la bibliothèque SDK ECW externe fournit par ERDAS.

A Problèmes de création

Le SDK ECW 4.x d'ERDAS est seulement gratuit pour la décompression d'image. Pour compresser des images il est nécessaire de compiler le SDK en lecture/écriture et de fournir une clé de licence OEM à son lancement qui peut être acheté chez ERDAS.

Pour ceux utilisant encore le SDK ECW 3.3, les images inférieures à 500 Mo peuvent être compressées grauitement, bien que les images plus grandes nécessite une licence de ERDAS. Voir l'agrément de la licence et l'option LARGE_OK.

Options de création :

- **TARGET=percent :** définie la taille cible de la réduction exprimé en pourcentage de l'originale. Si elle n'est pas fournit, la valeur par défaut est de 75 pour une réduction de 75 %. TARGET=0 utilise une compression sans perte.
- **PROJ=name :** nom de la projection ECW à utiliser. Des exemples employés sont NUTM11, ou GEODETIC.
- **DATUM=name :** nom du datum d'ECW à utiliser à utiliser. Des exemples employés sont WGS84 ou NAD83.
- **LARGE_OK=YES :** lorsque compilé avec le SDK ECW 3.x cette option peut être définie pour autoriser la compression des fichiers supérieur à 500 Mo. Il est de la responsabilité de l'utilisateur de s'assurer que les nécessités de licence pour la compression des gros fichiers a été suivit.
- **ECW_ENCODE_KEY=key :** fournie la clé d'encodage OEM acheté chez Erdas qui permet l'encodage des images. La clé a une longueur approximative de 129 hex. Il peut aussi être fournie comme option de configuration.

- **ECW_ENCODE_COMPANY=name :** fournie le nom de la société qui possède la clé d'encodage OEM d'ERDAS (voir ECW_ENCODE_KEY). Cela doit correspondre exactement au nom utilisé par ERDAS lors de la fourniture de la clé OEM. Il peut aussi être fournie comme option de configuration.
- **GMLJP2=YES/NO :** indique si une boîte GML conforme à OGC GML des spécifications du JPEG2000 doit être incluse dans le fichier. YES par défaut.
- **GeoJP2=YES/NO :** indique si une boîte GML conforme aux spécifications GeoJP2 (GeoTIFF dans JPEG2000) doit être incluse dans le fichier. YES par défaut.
- **PROFILE=profile :** Un profile parmi BASELINE_0, BASELINE_1, BASELINE_2, NPJE ou EPJE. Lisez la documentation du SDK d'ECW pour les détails sur la signification des profiles.
- **PROGRESSION=LRCP/RLCP/RPCL :** définie l'ordre de la progression avec quel codestream JPEG2000 il a été écrit.
- **CODESTREAM_ONLY=YES/NO :** si définie à YES, seulement le code stream de l'image compressée sera écrit. S'il est définie à NO (par défaut) un paquet JP2 sera écrit autour du code stream incluant divers meté information.
- **LEVELS=n** : Lire le SDK ECW pour les détails.
- **LAYERS=n** : Lire le SDK ECW pour les détails.
- **PRECINCT_WIDTH=n** : Lire le SDK ECW pour les détails.
- **PRECINCT_HEIGHT=n** : Lire le SDK ECW pour les détails.
- **TILE_WIDTH=n** : Lire le SDK ECW pour les détails.
- **TILE_HEIGHT=n** : Lire le SDK ECW pour les détails.
- **INCLUDE_SOP=YES/NO** : Lire le SDK ECW pour les détails.
- **INCLUDE_EPH=YES/NO** : Lire le SDK ECW pour les détails.
- **DECOMPRESS_LAYERS=n** : Lire le SDK ECW pour les détails.
- **DECOMPRESS_RECONSTRUCTION_PARAMETER=n** : Lire le SDK ECW pour les détails.

Le format JPEG2000 ne gère pas la création des aperçues puisque le format est déjà considéré comme suffisamment optimisé pour les « aperçues arbitraires ».

B Options de configuration

Le SDK ECW d'ERDAS gère une grande variété d'options de configuration pour contrôler différentes fonctionnalités. La plupart de celles-ci sont exposée par les options de configuration de GDAL. Voyez la documentation du SDK ECW pour plus de détails sur la signification de ces options.

- **ECW_CACHE_MAXMEM=bytes :** octets maximal de RAM utilisé pour la mise en cache mémoire. S'il n'est pas définie, jusqu'à 1/4 de la RAM physique sera utilisé par le SDK pour la mise en cache en mémoire.
- **ECW_TEXTURE_DITHER=TRUE/FALSE :** cela peut être définie à FALSE pour désactiver le tramage lors de la décompression des fichiers ECW. TRUE par défaut.
- **ECW_FORCE_FILE_REOPEN=TRUE/FALSE :** cela peut être définie à TRUE pour forcer à ouvrir un fichier pris en charge pour chaque fichier pour chaque connexion réalisée. FALSE par défaut.
- **ECW_CACHE_MAXOPEN=number :** le nombre maximal de fichier à garder ouvert pour que le fichier ECW prenne en charge la mise en cache. Illimité par défaut.
- **ECW_AUTOGEN_J2I=TRUE/FALSE :** Contrôle si les fichiers d'index .j2i doivent être créés lors de l'ouverture des fichiers jpeg2000. TRUE par édfaut.
- **ECW_RESILIENT_DECODING=TRUE/FALSE :** contrôle si le lecteur doit

oublier les erreurs dans un fichier et essayer de renvoyer autant de données que possible. TRUE par défaut. Si définie à FALSE un fichier invalide résultera en une erreur.

- **ECW_ENCODE_KEY, ECW_ENCODE_COMPANY :** ces valeurs, comme décrite dans les options de création, peuvent aussi être définir comme options de configuration. voir plus haut.

C Voir également

- Implémenté dans *gdal/frmts/ecw/ecwdataset.cpp*.
- SDK ECW disponible sur Erdas.com
- Astuces de compilation de l'ECW pour GDAL

Chapitre XXXVIII JP2KAK -- JPEG-2000 (basé sur Kakadu)

La plupart des formes des images compressées JPEG2000 JP2 et JPC (ISO/IEC 15444-1) peuvent être lu avec GDAL utilisant le pilote basé sur la bibliothèque Kakadu. De même, de nouvelles images peuvent être écrites. Les images existantes ne peuvent pas être mises à jour.

Le format de fichier JPEG2000 gère les compressions avec et sans perte d'image 8 et 16 bits avec une ou plusieurs bandes (composants). Via le mécanisme GeoJP2 (http://www.mappingscience.com/msi.html), le système de coordonnées et les informations de géoréférencement style GeoTIFF peuvent être inclus dans le fichier JP2. Les fichiers JPEG2000 utilisent un format et un mécanisme de compression substantiellement différent de la compression JPEG traditionnelle et du format JPEG JFIF. Il y a des mécanismes de compressions distincts produit par le même groupe. JPEG2000 est basé sur la compression par ondelette.

Le pilote JPEG200 documenté sur cette page (le pilote JP2KAK) est implémenté par-dessus de la bibliothèque commerciale Kakadu (http://www.kakadusoftware.com/). C'est une bibliothèque JPEG200 de grande qualité et hautement performante, largement utilisée dans la communauté géomatique et dans l'imagerie générale. Cependant, elle n'est pas libre et donc une compilation normale de GDAL à partir des sources n'inclura pas un support pour ce pilote à moins que le compilateur achète une licence pour la bibliothèque et configure en conséquence. GDAL inclut un autre pilote JPEG200 basé sur la bibliothèque libre JasPer (http://www.gdal.org/frmt_jpeg2000.html).

Lors de la lecture d'une image ce pilote représentera les bandes en byte (8 bits non signés), 16 bite signé ou 16 bits non signé. Les informations du système de coordonnées et de géoréférencement seront disponibles si le fichier est un fichier GeoJP2 (tm). Les fichiers couleurs encodés dans l'espace de couleur YCbCr seront automatiquement traduit en RVB. Les images avec une palette sont également gérées.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraite des fichiers JPEG2000, et seront stockées comme contenu brute XML dans le domaine de métadonnées xml:XMP.

A Problèmes de création

Les fichiers JPEG2000 peuvent seulement être crée en utilisant le mécanisme *CreateCopy* pour les copier dans un jeu de donnée existant.

Les aperçues JPEG2000 sont maintenu comme partie de la description mathématique de l'image. Les aperçues ne peuvent pas être construit comme un traitement séparé, mais en lecture l'image sera généralement représenté comme ayant des niveaux d'aperçus à différentes puissances de deux facteurs.

Options de création :

- **QUALITY=n** : Définie le taux de la taille compressée comme un pourcentage de la taille de l'image non compressée. Par défaut elle est de 20 et indique que l'image doit avoir une taille de 20 % de celle de l'image non compressée. La taille de l'image finale réelle peut ne pas correspondre exactement celle demandée en fonction de différents facteurs. Une valeur de 100 entraînera l'utilisation d'un algorithme de compression sans perte. Sur les données images typique, si vous définissez une valeur plus grande que 65, il peut être intéressant de tester avec *QUALITY=100* puisque la compression sans perte peut produire une meilleure compression qu'une compression sans perte.
- **BLOCKXSIZE=n** : Définie la largeur de la tuile à utiliser. Par défaut elle est de 20000.
- **BLOCKYSIZE=n** : Définie la hauteur de la tuile à utiliser. Par défaut elle est de la hauteur de l'image.
- **GMLJP2=YES/NO** : Indique si une boîte GML conforme au GML de l'OGC dans les spécification JPEG2000 doit être inclus dans le fichier. Oui par défaut.
- **GeoJP2=YES/NO** : Indique si une boîte GML conforme aux spécifications GeoJP2 (GeoTIFF dans JPEG2000) doit être inclus dans le fichier. Oui par défaut.
- **LAYERS=n** : Contrôle le nombre de couche produite. C'est une sorte de résolution de couches, mais pas tout à fait. La valeur par défaut est de 12 et cela fonctionne bine dans la plupart des cas. ROI=xoff,yoff,xsize,ysize : Sélectionne une région pour être une région d'intérêt pour faire des traitements avec des données de meilleurs qualités. Les différents options « R » ci-dessous peuvent être utilisé pour mieux contrôler le taux. Par exemple, les définitions "ROI=0,0,100,100", "Rweight=7" encoderaient la zone de 100x100 en haut à gauche de l'image avec une qualité considérablement plus élevée comparée au reste de l'image. Les options de création suivantes sont hautement lié à la bibliothèque Kakadu, et sont donnée pour une utilisation avancée seulement. Consultez la documentation de Kakadu pour une meilleur compréhension de leur signification.
- **Corder** : Défaut à "PRCL".
- **Cprecincts** : défaut à "{512,512},{256,512},{128,512},{64,512},{32,512}, {16,512},{8,512},{4,512},{2,512}".
- **ORGgen_plt** : défaut à "yes".
- **Cmodes** : défaut de la bibliothèque Kakadu utilisé.
- **Clevels** : défaut de la bibliothèque Kakadu utilisé.
- **Rshift** : défaut de la bibliothèque Kakadu utilisé.
- **Rlevels** : défaut de la bibliothèque Kakadu utilisé.
- **Rweight** : défaut de la bibliothèque Kakadu utilisé.

Lire également :

- Implémenté dans *gdal/frmts/jp2kak/jp2kakdataset.cpp*.
- JPEG2000 pour la pag Applications Geospatial, inclus la discussion GeoJP2(tm) :

http://www.remotesensing.org/jpeg2000/.

- Pilote JPEG200 alternatif : http://www.gdal.org/frmt_jpeg2000.html.

Chapitre XXXIX JP2MrSID --- JPEG2000 via MrSID SDK

Le format de fichier JPEG2000 est géré en lecture avec le DSDK MrSID. Il gère également l'écriture avec l'ESDK MrSID.

La gestion du JPEG2000 de MrSID est seulement disponible avec la version 5.X ou plus récente du DSDK et du EDSK.

A Options de création

Si vous avez l'ESDK de MrSID (5.X ou plus récent), il peut être utilisé pour écrire des fichiers JPEG2000. Les options de création suivantes sont gérées :

- **WORLDFILE=YES** : pour écrire un fichier world ESRI (avec l'extension .j2w).
- **COMPRESSION=n** : indique le taux de compression désiré. Zéro indique un taux de compression de 20:1 (l'image sera compressée au 1/20 de la taille originale).
- **XMLPROFILE=[chemin vers le fichier]** : indique un chemin au profile XML spécifique de LizardTech qui peut être utilisé pour paramétrer l'encodage en JPEG2000. Ils peuvent être créer en utilisant l'ESDK de MrSID, ou avec GeoExpress, ou à la main en utilisant l'exemple suivant comme modèle :

```
<?xml version="1.0"?>
<Jp2Profile version="1.0">
    <Header>
        <name>Default</name>
        <description>LizardTech preferred settings
(20051216)</description>
    </Header>
    <Codestream>
        <layers>
            8
        </layers>
        <levels>
            99
        </levels>
        <tileSize>
            0 0
```

```
        </tileSize>
        <progressionOrder>
            RPCL
        </progressionOrder>
        <codeblockSize>
            64 64
        </codeblockSize>
        <pltMarkers>
            true
        </pltMarkers>
        <wavelet97>
            false
        </wavelet97>
        <precinctSize>
            256 256
        </precinctSize>
    </Codestream>
</Jp2Profile>
```

Lire également :

- Implémenté dans *gdal/frmts/mrsid/mrsiddataset.cpp*.
- Site de LizardTech : http://www.lizardtech.com/

Chapitre XL JPEG -- JPEG JFIF File Format

Le format JPEG JFIF est géré en lecture et écriture batch, mais ne met pas à jour. Les fichiers JPEG sont représentés comme des jeux de données d'une bande (nuance de gris) ou de trois bandes (RVB) avec des bandes de valeurs en byte.

Le pilote convertira automatiquement les images dont l'espace de couleur est YCbCr, CMYK ou YCbCrK vers RVB à moins que l'option *GDAL_JPEG_TO_RGB* est définie à *NO* (*YES* par défaut). Lorsque la traduction de l'espace de couleur vers RVB est réalisée, l'espace de couleur source est indiquée dans les méta-données *SOURCE_COLOR_SPACE* du domaine *IMAGE_STRUCTURE*.

Il n'y a actuellement aucune gestion pour les informations de géo-référencement ou des méta-données pour les fichiers JPEG. Mais si un fichier world ESRI existe avec l'extension .jgw, .jpgw/.jpegw ou .wld comme suffixes, il sera lu et utilisé pour établir la géo-transformation pour l'image. S'il est disponible un fichier MapInfo .tab sera également utilisé pour le géoréférencement. Les aperçues peuvent être construit pour les fichiers JPEG dans un fichier .ovr externe.

Duplicate explicit target name: "rfc 15".

Le pilote gère également l'approche "ajout au fichier du masque compressé zlib" utilisé par quelques fournisseurs de données pour ajouter un masque pour identifier les pixels qui ne sont pas des données valides. Voir RFC 15 pour plus de détails.

Le pilote JPEG de GDAL est compilé en utilisant la bibliothèque jpeg du *Groupe JPEG indépendant*. Notez également que le pilote GeoTIFF gère les TIFF tuilés avec des tuiles compressées en JPEG.

Pour lire et écrire des images JPEG avec des échantillons de 12 bits, vous pouvez compiler GDAL avec la bibliothèque libjpeg interne (basé sur IJG libjpeg-6b, avec des modifications supplémentaires pour la gestion de l'échantillon 12-bit), ou passer explicitement *--with-jpeg12=yes* au script configure lors de la compilation externe de libjpeg. Voir la page wiki "8 and 12 bit JPEG in TIFF" pour plus de détails.

Il est également possible d'utiliser le pilote JPEG avec libjpeg-turbo, une version de libjpeg, compatible API et ABI avec IJG libjpeg-6b, qui utilise les instructions MMX, SSE et SSE2 SIMD pour accélérer la compression/décompression du JPEG baseline.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraite des fichiers, et seront stockées comme contenu brute XML dans le domaine de métadonnées xml:XMP.

A Options de création

Les fichiers JPEG sont crée en utilisant le code du pilote « JPEG ». Seules les types de bande en byte sont gérés, et seulement les configurations 1 et 3 bandes (RGB). La création des fichiers JPEG est implémentée par la méthode batch (CreateCopy). Les espaces de couleurs YCbCr, CMYK ou YCbCrK ne sont pas gérés en création. Si le jeu de données source possède un masque nodata, il sera ajouté comme un masque compressé en zlib au fichier JPEG :

- **WORLDFILE=YES :** Force la génération d'un fichier world ESRI associé (avec l'extension .wld).
- **QUALITY=n :** Par défaut l'option qualité est défini à 75, mais cette option peut être utilisée pour définir une autre valeur. La valeur doit être compris entre 10 et 100. De faibles valeurs entraînent une plus grand taux de compression, mais une qualité d'image moins bonne. Les valeurs au-dessus de 95 ne sont pas significativement de meilleur qualité mais peuvent être substantiellement plus importante.
- **PROGRESSIVE=ON :** Active la génération de jpeg progressif. Dans certain cas cela affichera une image d'une résolution réduite dans un visualiseur tel que Netscape et Internet Explorer, avant que le fichier entier ne soit téléchargé. Cependant, certaines applications ne peuvent pas lire les jpeg progressive du tout. GDAL peut lire les jpeg progressifs, mais n'utilise pas l'avantage de leur nature progressive.

1 Voir également

- Indépendant JPEG Group : http://www.ijg.org/
- libjpeg-turbo
- :ref:`gdal.gdal.formats.gtiff`
  Unknown interpreted text role "ref".

Chapitre XLI JPEG2000 --- Implémentation du format JPEG-2000 partie 1

Cette implémentation est basé sur le logiciel JasPer (voir ci-dessous).

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraite des fichiers, et seront stockées comme contenu brute XML dans le domaine de métadonnées xml:XMP.

A Options de création

- **WORLDFILE=ON** : Force la génération d'un fichier world associé (.wld).
- **FORMAT=JP2|JPC** : Définie le format du fichier en sortie.
- Paramètres d'encodage, directement distribué à la bibliothèque JasPer décrit dans la documentation JasPer.

Citation de la documentation :

Les options suivantes sont gérées par l'encodeur :

imgareatlx=x	Définie les coordonnées x du coin haut gauche de la zone d'image à x.
imgareatly=y	Définie les coordonnées y du coin haut gauche de la zone d'image à y.
tilegrdtlx=x	Définie les coordonnées x du coin haut gauche de la grille de tuile à x.
tilegrdtly=y	Définie les coordonnées y du coin haut gauche de la grille de tuile à y.
tilewidth=w	Définie la largeur de la tuile nominale à w.
tileheight=h	Définie la hauteur de la tuile nominale à h.
prcwidth=w	Définie la largeur du pourtour à w. L'argument w doit être un entier de puissance deux. La valeur par défaut est de 32768.
prcheight=h	Définie la hauteur du pourtour à h. L'argument h doit être un entier

	de puissance deux. La valeur par défaut est de 32768.
cblkwidth=w	Définie la largeur du bloque de code nominale à w. L'argument w doit être un entier de puissance de deux. La valeur par défaut est de 64.
cblkheight=h	Définie la hauteur du bloque de code nominale à h. L'argument h doit être un entier de puissance de deux. La valeur par défaut est de 64.
mode=m	Définie le mode de codage à m. L'argument m doit avoir une des valeurs :ref:`gdal.gdal.formats.jpeg2000.argument_m`. Si un codage sans perte est désirée, le mode entier doit être utilisé. Par défaut, le mode entier est utilisé. Le choix du mode détermine également quel multicomposant et transformation par ondelette (si existante) sont employé. Unknown interpreted text role "ref".
rate=r	Définie le taux cible. L'argument r est un nombre réel positif. Puisque un taux de un correspond à aucune compression, vous ne devez pas définir explicitement un taux plus grand que un. Par défaut, le taux cible est considéré comme étant infinie.
ilyrrates=[, ,. . . ,]	Définie les taux pour n'importe quelles couches intermédiaires. L'argument de cette option est une liste séparé par des virgules de N taux. Chaque taux est un nombre réel positif. Les taux doivent augmenter graduellement. Le dernier taux dans la liste doit être inférieure ou égale à le taux d'ensemble (comme définie avec l'option rate).
prg=p	Définie l'ordre de progression à p. L'argument p doit avoir une des valeurs :ref:`gdal.gdal.formats.jpeg2000.argument_p`. Par défaut, l'ordre progressif LRCP est employé. Remarquez que les progressions RPCL et PCRL ne sont pas valide pour toute les géométries des images possibles (Lisez les standards pour plus de détails). Unknown interpreted text role "ref".
nomct	Désactive l'utilisation de n'importe quelle transformation multicomponent.
numrlvls=n	Définie le nombre de niveaux de résolution à n. L'argument n doit être un entier qui est plus grand ou égale à un. La valeur par défaut est 6.
sop	Génère des segments de marqueurs SOP.
eph	Génère des segments de marqueurs EPH.
lazy	Active le mode de codage fainéant (autrement dit dérivation de codage arithmétique).
termall	Termine toutes les passes de codage.

segsym	Utilise les symboles de segmentation.
vcausal	Utilise les contextes de causalité de bande vertical.
pterm	Utilise les terminaisons prédictifs.
resetprob	Réinitialise les modèles de probabilité après chaque passage de codage.
numgbits=n	Définie le nombre de bits de garde à n.

1 Argument m

Valeur	*Description*
int	mode entier
real	mode réel

2 Argument P

Valeur	*Description*
lrcp	layer-resolution-component-position (LRCP) progressive (c'est à dire, évolution du taux)
rlcp	resolution-layer-component-position (RLCP) progressive (c'est à dire, évolution de la résolution)
rpcl	resolution-position-component-layer (RPCL) progressive
pcrl	position-component-resolution-layer (PCRL) progressive
cprl	component-position-resolution-layer (CPRL) progressive

Voir également :

- Implémenté dans *gdal/frmts/jpeg2000/jpeg2000dataset.cpp*.
- Vous avez besoin de la bibliothèque JasPer pour compiler ce pilote avec la gestion du format GeoJP2 activé. La version modifiée peut être téléchargée sur ftp://ftp.remotesensing.org/gdal/jasper-1.900.1.uuid.tar.gz
- Page officielle du format JPEG-2000 : http://www.jpeg.org/JPEG2000.html
- La page du projet JasPer : http://www.ece.uvic.ca/~mdadams/jasper/

Autre pilote GDAL JPEG2000 :

- :ref:`gdal.gdal.formats.jp2openjpeg`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2ecw`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2mrsid`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2kak`
  Unknown interpreted text role "ref".

Chapitre XLII JP2OpenJPEG --- pilote JPEG2000 basé sur la bibliothèque OpenJPEG

(GDAL >= 1.8.0)

Ce pilote est une implémentation d'un lecteur/écriture de JPEG2000 basé sur la bibliothèque OpenJPEG **v2**.

La branche v2 n'est - au moment de la rédaction - pas encore publiée, vous devez donc la récupérer nous même à partir du dépôt Subversion : http://openjpeg.googlecode.com/svn/branches/v2. Le trunk OpenJPEG est encore dérivé de la série 1.3 qui n'apporte pas l'accès à la lecture des niveaux de tuiles. La branche v2 active la lecture des grosses images JPEG2000 sans les charger complètement en mémoire. C'est une amélioration visible en comparaison du pilote JPEG2000 basé sur la bibliothèque Jasper.

La branche v2 ajoute également la possiblité d'utiliser l'API VSI Virtual File, le pilote peut donc lire les fichiers NITF compressés en JPEG2000.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraites des fichiers JPEG2000, et seront stockées comme contenu brute XML dans le domaine de métadonnées xml:XMP.

En création, le piloet ne gère pas l'écriture de GeoJP2 ou GMLJP2.

A Options de création

- **CODEC=JP2/J2K :** JP2 ajoutera des boîtes JP2 autour des données codestream. La valeur est déterminé automatiquement à partir de l'extension du fichier. Si c'est ni JP2 ou J2K, le codec J2K sera utilisé.
- **QUALITY :** pourcentage entre 0 et 100. Une valeur de 50 signifie que le fichier sera de moité de taille en comparaison aux données non compressées, 33 signifie 1/3, etc.. 25 par défaut.
- **REVERSIBLE=YES/NO :** YES signifie compression sans perte. No par défaut.
- **RESOLUTIONS :** Nombre de niveaux de résolution. Entre 1 et 7. 6 par défaut.
- **BLOCKXSIZE :** largeur de la tuile . 1024 par défaut.
- **BLOCKYSIZE :** hauteur de la tuile. 1024 par défaut.

- **PROGRESSION :** ordre de progression : LRCP, RLCP, RPCL, PCRL ou CPRL. LRCP par défaut.
- **SOP=YES/NO :** YES signifie générer des segments marqueur SOP. No par défaut.
- **EPH=YES/NO :** YES signifie générer des segments marqueur EPH. No par défaut.

B Patches pour la bibliothèque OpenJPEG

Liens vers des patches utiles à appliquer à OpenJPEG (valide pour la branche v2 à la révision r565) :

- Fixe une faiblesse dans jp2_read_header_procedure()
- Fixe un conflit de noms de fonction interne avec Jaspe qui peuvent entraîner des crashes

C Voir également

- Implémenté dans *gdal/frmts/openjpeg/openjpegdataset.cpp*.
- Page official du JPEG-2000
- La page de la bibliothèque OpenJPEG

Autres pilotes JPEG2000 pour GDAL :

- :ref:`gdal.gdal.formats.jpeg2000`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2ecw`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2mrsid`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.jp2kak`
  Unknown interpreted text role "ref".

Chapitre XLIII JPEGLS

(GDAL >= 1.8.0)

Ce pilote est une implémentation d'un lecteur/écriture de JPEG-LS basé sur la bibliothèque Open Source CharLS (Licence style BSD). Au moment de la rédaction, la bibliothèque CharLS dans sa version 1.0 n'a pas de cible "`make install`". Par conséquence, l'intégration du pilote dans le système de compilation de GDAL est un peu rude. Sur Unix, vous devez éditer le fichier *GDALmake.opt* et éditer les lignes liées à CHARLS.

Le pilote peut lire et écrire des images sans perte ou proche du sans perte. Notez qu'il ne vise pas à traiter des images trop grandes (sauf si virtual memory est disponible), puisque l'ensemble de l'image doit être compressé/décompressé en une seule opération.

A Options de création

- **INTERLEAVE=PIXEL/LINE/BAND :** entrelacement des données en flux compressé. BAND par défaut.
- **LOSS_FACTOR=error_threshold :** 0 (la valeur par défaut) signifie une compression sans perte. N'importe quelle valeur plus haute sera la limite maximale pour l'erreur.

B Voir également

- Implémenté dans *gdal/frmts/jpegls/jpeglsdataset.cpp*.
- Page principale de la bibliothèque CharLS

Chapitre XLIV JPIPKAK - JPIP Streaming

JPEG 2000 Interactive Protocol (JPIP) flexibilité en ce qui concerne l'accès aléatoire, ordonnancement du flux du code et décodage incrémental est hautement exploitable dans un environnement réseau permettant l'accès aux gros fichiers distants avec une bande passante des connexions limitée ou des réseaux à haute concurrence.

A JPIPKAK - aperçu JPIP

Un bref aperçu de la séquence des événements JPIP est présenté dans cette section, plus d'information peut être trouvé dans JPEG 2000 Interactive Protocol (Part 9 - JPIP) et la spécification peut (et doit) être acheté à partir du site de l'ISO.

Une version antérieure du JPEG 2000 partie 9 est disponible ici http://www.jpeg.org/public/fcd15444-9v2.pdf, notez le copyright ISO, les diagrammes ne sont pas répliqués dans cette documentation.

Une abstraction du protocole JPIP a été réalisé dans ce pilote de format, les requêtes ont été réalisé à une niveau de résolution 1:1.

1. la demande JPIP initiale pour une image cible, un id cible, un session sur http, les données devant être renvoyées comme jpp-stream sont demandé et une longueur maximale est placé dans la réponse. Dans ce cas aucune fenêtre initiale est demandée, bien que cela peut l'être. Le serveur répond avec un identifiant cible qui peut être utilisé pour identifier l'image sur le serveur et en-tête de réponse JPIP-Cnew qui inclut le chemin vers le serveur JPIP qui va traiter toutes les demandes à venir et l'identifiant cid de la session. Une session est nécessaire afin que que le serveur puisse modéliser l'état de la connexion du client, envoi seulement les données nécessaire.
2. Le client demande des fenêtres de vue particulière sur l'image cible avec une longueur de réponse maximale et comprend l'identifiant de session établie dans la communication précédente . 'fsiz' est utilisé pour identifier la résolution associée avec la demande de la fenêtre de vue. Les valeurs 'fx' et 'fy' définissent les dimensions de la résolution de l'image désirée. 'roff' est utilisé pour identifier le coin supérieur gauche en dehors de la région spatiale associée à la fenêtre de vue demandée. 'rsiz' est utilisé pour identifier l'étendue horizontal et vertical de la région spatiale associée à la fenêtre de vue demandée.

B JPIPKAK - approche

Le pilote JPIPKAK utilise une approche qui a d'abord été démontré ici, J2KViewer, par Juan Pablo Garcia Ortiz en séparant la couche de communication (socket / http) à partir de l'objet kdu_cache Kakadu. Séparer la couche de communication de l'objet données est désirable puiqu'elle permet l'utilisation de bibliothèques client http optimisée tels que libcurl, Apache HttpClient (notez que jportiz utilise un Java complet) et permet la communication SSL entre le client et le serveur.

L'implémentation Kakadu de la communication du client avec un serveur JPIP utilise un socket, et cette connexion du socket détient l'état de cette session du client. Une session client avec Kakadu peut être recréer en utilisant les opérations de cache de JPIP entre le client et le serveur, mais aucune utilisation des cookies HTTP traditionnel est géré puisque JPIP est neutre pour la couche transport.

Le pilote JPIPKAK est écrit en utilisant une bibliothèque client HTTP avec l'objet cache de Kakadu et la gestion de la communication optimisée ave un serveur JPIP (qui peut ou non gérer les sessions HTTP) et le kdu_region_decompressor haute performance de Kakadu.

C JPIPKAK - implémentation

L'implémentation gère les API C et C++ de GDAL et fournie un wrapper SWIG initial pour ce pilote avec un exemple ImageIO Java (**TODO** - Exemple qGIS).

Le pilote utilise un modèle simple de threading pour gérer les requêtes de lecture des données et la récupération distance. Ce modèle de threading gère deux fenêtres clientes séparées, avec juste une connexion sur le serveur. Les requêtes vers le serveur sont multiplexées pour utiliser la bande passant disponible efficacement. Le client identifie ces fenêtres en utilisant les valeurs "0" (faible) ou "1" (haut) pour une option de la requête de métadonnées "PRIORITY".

Note : gestion SSL

Si le client est compilé avec la gestion de SSL, alors le pilote détermine s'il utilise SSL si la requête est un protocole jpips:// en opposition à jpip://. Notez que le pilote ne vérifie pas les certificats des serveurs en utilisant le bundle du certificat de Curl et il est pour le moment définir pour accepter tous les certificats du serveur SSL.

Note : libCurl

JPIP définie des valeurs client/serveur en utilisant les en-têtes HTTP, les modifications ont été faire dans la bibliothèque de portabilité HTTP de GDAL pour gérer cela.

1. *GDALGetDatasetDriver*
   récupère le pilote auquel ce jeu de données est lié.
2. *Open*
   Si le nom du fichier contenu dans l'objet `GDALOpenInfo` a un schéma d'URI insensible à la casse de jpip ou jpips le `JPIPKAKDataset` est créé et initialisé, autrement NULL est retourné.
3. *Initialize*
   L'initialisation implique de faire une connexion initiale au serveur JPIP pour établir une session et pour récupérer les métadonnées initiales sur l'image (ref. :ref:`gdal.gdal.formats.jpip.sequence`).
   Unknown interpreted text role "ref".
   si la connexion échoue, la fonction renvoie false et la fonction `Open` retourne

NULL indiquant que l'ouverture du jeu de données avec ce pilote a échouée.
Si la connexion est réussie, alors les demandes suivantes au serveur JPIP sont réalisé pour récupérer toutes les métadonnées disponibles des images. Les items des métadonnées sont définie en utilisant `GDALMajorObject->SetMetadataItem` dans le domanie "JPIP".
Si la métadonnée renvoyée du serveur inclus la boîte GeoJP2 UUID, ou une boîte XML GMLJP2 alors cette métadonnées est parsée et définie la métadonnées géographique de ce jeu de données.

4. *GDALGetMetadata*
   API C vers `JPIPKAKDataset->GetMetadata`
5. *GetMetadata*
   retourne les métadonnées pour le domaine "JPIP", les clés sont "JPIP_NQUALITYLAYERS", "JPIP_NRESOLUTIONLEVELS", "JPIP_NCOMPS" et "JPIP_SPRECISION"
6. *GDALEndAsyncRasterIO*
   si le raster IO asynchrone est actif et pas nécessaire, l'API C appelle `JPIPKAKDataset->EndAsyncRasterIO`
7. *EndAsyncRasterIO*
   L'objet JPIPKAKAsyncRasterIO est supprimé
8. *delete*
9. *GDALBeginAsyncRasterIO*
   API C vers `JPIPKAKDataset->BeginAsyncRasterIO`
10. *BeginAsyncRasterIO*
    le client a définie la fenêtre de visualisation demandée à 1:1 et à définie optionnellement les items des métadonnées du niveau d'annulation, la qualité des couches et la priorité des threads.
11. *Create*
    créé un objet JPIPKAKAsyncRasterIO
12. *Start*
    Configure la machinerie de kakadu et démarre un thread d'arrière-plan (s'il ne fonctionne pas déjà) pour communiquer au serveur la demande de la fenêtre de visualisation acutelle. Le thread d'arrière-plan résulte dans la mise à jour de l'objet `kdu_cache` tant que le serveur JPIP envoie un message "End Of Response" (EOR) pour la demande de fenêtre de visualisation actuelle.
13. *API C vers LockBuffer*
14. *LockBuffer*
    Non implémented dans `JPIPKAKAsyncRasterIO`, un blocage est nécessaire dans `JPIPKAKAsyncRasterIO->GetNextUpdatedRegion`
15. *GDALGetNextUpdatedRegion*
    C API to GetNextUpdatedRegion
16. *GetNextUpdatedRegion*
    La fonction décompresse les données disponibles pour générer une image (en fonction du type de buffer du jeu de données définie dans `JPIPKAKDataset->BeginAsyncRasterIO`). La largeur et la hauteur de la fenêtre (au niveau de rejet demandée) décompressée est renvoyée dans la pointeur de la région et peut être affichée par le client. Le statut de l'opération de rendue est une parmi `GARIO_PENDING, GARIO_UPDATE, GARIO_ERROR, GARIO_COMPLETE` à partir de la structure `GDALAsyncStatusType`. `GARIO_UPDATE, GARIO_PENDING` nécessite plusieurs lectures de GetNextUpdatedRegion pour obtenir les données image complète, ceci est le rendu progressif de JPIP. `GARIO_COMPLETE` indique que la fenêtre est complète.
    `GDALAsyncStatusType` est une structure utilisée par `GetNextUpdatedRegion`

pour indiquer si la fonction doit être appelée encore lorsque soit kakadu a plus de données dans son cache à décompresser, ou le serveur n'a pas envoyé un message End Of Response (EOR) pour indiquer que la fenêtre demandée est complète.
La région passée dans cette fonction est envoyée par référence, et *l'appelleur peut lire cette région quand le résultat renvoie pour trouver la région qui a été décompressée ([NdT] phrase en anglais peu clair)*. Les données images sont placées dans le buffer, par exemple RGB si la région demandée possède 3 composants.

17. *GDALUnlockBuffer*
    Api C vers UnlockBuffer
18. *UnlockBuffer*
    Non implémenté dans `JPIPKAKAsyncRasterIO`, un blocage est acquis dans `JPIPKAKAsyncRasterIO->GetNextUpdatedRegion`
19. *Draw*
    Client réalise le rendu des données image
20. *GDALLockBuffer*
21. *LockBuffer*
22. *GDALGetNextUpdatedRegion*
23. *GetNextUpdatedRegion*
24. *GDALUnlockBuffer*
25. *UnlockBuffer*
26. *Draw*

D JPIPKAK - exigences d'installation

- Libcurl 7.9.4
- OpenSSL 0.9.8K (si SSL est nécessaire, une connexion JPIPS)
- Kakadu (testé avec v5.2.6 et v6)

Pour le moment seulement un makefile Windows est fournie, cependant cela devrait compiler sous Linux également puisqu'il n'y a pas de dépendances Windows.

Voir également :

- JPEG 2000 Interactive Protocol (Part 9 - JPIP)
- http://www.opengeospatial.org/standards/gmljp2
- Kakadu Software
- IAS demo (example JPIP(S) streams)

E Notes

Pilote développé originellement par <http://www.ittvis.com">ITT VIS et donné à GDAL pour permettre le flux de jeux de données JPEG 2000 de client JPIP avec le SSL activé .

Chapitre XLV L1B -- NOAA Polar Orbiter Level 1b Data Set (AVHRR)

GDAL gère le format NOAA Polar Orbiter Level 1b Data Set en lecture. Il peut lire maintenant les jeux de données NOAA-9(F) --- NOAA-17(M). Remarque : seul ces instruments AVHRR sont gérés pour l'instant, si vous désirez lire des données provenant d'autres instruments écrivez moi (Andrey Kiselev, dron@ak4719.spb.edu). AVHRR LAC/HRPT (résolution de 1 km) et GAC (résolution de 4 km) devraient être traité correctement.

A Géo-référencement

Notez que le modèle de géoréférencement par affine simple de GDAL est complètement inutilisable pour les données NOAA. Vous ne devez pas le relier. Il est recommandé d'utiliser le *wrapper thin plane spline* (tps). La rectification automatique d'image peut être réalisé avec des points d'amer (GCP en anglais) à partir du fichier d'entrée. Le format NOAA range 51 points d'amer par ligne scannée à la fois dans les jeux de données LAC et GAC. En fait, vous pouvez avoir moins de 51 points d'amer, spécialement à la fin des lignes scannées. Une autre approche pour la rectification est la sélection manuelle des points d'amer en utilisant des sources externes d'information de géoréférencement.

La précision de la détermination des points d'amer dépend du type de satellite. Dans les jeux de données NOAA-9 – NOAA-14 les coordonnées géographiques des points d'amer sont rangés sous la forme de valeur entière eau 128 ème de degré. Nous ne pouvions pas déterminer les positions plus précisément que 1/28 = 0,0078125 de NOAA-15 – NOAA-17 nous avons beaucoup plus de position précise, ceux-ci sont rangés sous forme de 10 000 ème de degré.

Les images seront toujours retournées avec la ligne scannée la plus au nord localisée en haut de l'image. Si vous désirez déterminer la direction réelle du déplacement du satellite vous devez regarder la méta-données LOCATION.

B Données

Dans le cas du NOAA-10 dans le canal 5 vous obtiendrez la répétition du canal 4 des données. Les instruments AVHRR/3 (NOAA-15 – NOAA-17) est un radiomètre à six canaux, mais seulement cinq canaux sont transmis au sol à n'importe quel moment. Les

canaux 3A et 3B ne peuvent pas être utilisé au même moment. Regardez le champ description du canal reporté par `gdalinfo` pour déterminer quel sorte de canal est contenu dans le fichier de traitement.

C Méta-données

Plusieurs paramètres, obtenu à partir du jeu de données sont rangés comme enregistrement de méta-données.

Les enregistrements des méta-données :

- **SATELLITE :** nom du satellite
- **DATA_TYPE :** type de la donnée, rangée dans le niveau 1b du jeu de données (AVHRR HRPT/LAC/GAC).
- **REVOLUTION :** numéro de l'orbite. Notez que cela peut être décalé d'un ou de deux numéro de l'orbite correcte (selon la documentation).
- **SOURCE :** nom de la station réceptrice.
- **PROCESSING_CENTER :** nom du centre de traitement de la données.
- **START :** heure du début de l'acquisition de la ligne scannée (année, jour de l'année, milliseconde du jour).
- **STOP :** heure de la fin de l'acquisition de la ligne scannée (année, jour de l'année, milliseconde du jour).
- **LOCATION :** indication de la localisation par rapport à la terre de l'AVHRR (AVHRR Earth location indication). Ce sera soit *Ascending* quand le satellite se déplace des faibles latitudes vers les hautes latitudes, et *Descending* pour les autres cas.

Lisez également :

- Implémenté dans gdal/frmts/l1b/l1bdataset.cpp.
- NOAA Polar Orbiter Level 1b Data Set est documenté dans `POD User's Guide` (TIROS-N -- NOAA-14 satellites) et dans `NOAA KLM User's Guide` (NOAA-15 -- NOAA-16 satellites). Vous pouvez trouver ces manuels sur la page d'introduction de la documentation technique de laNOAA : http://www2.ncdc.noaa.gov/docs/intro.htm
- un excellent et complet rapport est contenu dans le livre imprimé `The Advanced Very High Resolution Radiometer (AVHRR)` par Arthur P. Cracknell, Taylor et Francis Ltd., 1997, ISBN 0-7484-0209-8.
- des données NOAA peuvent être téléchargées à partir du site Comprehensive Large Array-data Stewardship System (CLASS) : http://www.class.noaa.gov/ (anciennement SAA). En réalité ce sont des jeux de données de niveau 1B selon l'auteur du pilote, son implémentation a été testé seulement avec ces fichiers.
- page de statuts des brouillons de la NOAA : http://www.oso.noaa.gov/poesstatus/
- http://www.lizardtech.com/

Chapitre XLVI Pilote GDAL pour le format FARSITE v.4 LCP

Les fichiers FARSITE v. 4 landscape (LCP) est un format raster multi-bande utilisé par modélisations de simulation de comportement des incendies et des effets du feu tels que FARSITE, FLAMMAP et FBAT (www.fire.org). Les bandes d'un fichier LCP stocke des données décrivant le terrain, la canopée des arbres, et la surface du pétrole. L'USGS National Map for LANDFIRE distribue des données au format LCP, et des programmes tels que FARSITE et LFDAT peuvent créer des fichiers LCP à partir de raster en entré. Le pilote GDAL pour LCP gère la lecteur seule.

Un fichier LCP (.lcp) est essentiellement un format brute avec un en-tête de 7,316-byte décrit ci-dessous. Le type de données pour toutes les bandes est un entier de 16 bit signé. Les bandes sont entrelacées par pixel. Cinq bandes sont nécessaire : élévation, pente, aspect, modèle de pétrole, et couverture de la canopée des végétaux. Les bandes de *crown fuel* (hauteur de la canopée, la hauteur du couvert de base, la densité du couvert), et de la surface des bandes de pétrole (humus, les débris ligneux grossiers) sont optionnels.

Le pilote LCP lit les unités linéaire, la taille de la cellule, et l'étendue mais le fichier LCP ne définie pas la projection. Les projections UTM sont typique mais d'autres projections sont possible.

Le pilote LCP de GDAL renvoie des informations sur les méta-données des jeux de données et les niveaux de bandes :

Jeu de données

```
LATITUDE: Latitude of the dataset, negative for southern hemisphere
LINEAR_UNIT: Feet or meters
DESCRIPTION: LCP file description
```

Bande

```
<band>_UNIT or <band>_OPTION: units or options code for the band
<band>_UNIT_NAME or <band>_OPTION_DESC: descriptive name of
units/options
```

```
<band>_MIN: minimum value
<band>_MAX: maximum value
<band>_NUM_CLASSES: number of classes, -1 if > 100
<band>_VALUES: comma-delimited list of class values (fuel model band
only)
<band>_FILE: original input raster file name for the band
```

Note!

Le pilote LCP dérive de la classe helper RawDataset déclarée dans gdal/frmts/raw. Il doit être implémenté dans gdal/frmts/raw/lcpdataset.cpp.

Format d'en-tête LCP :

byte de début	***N°. de bytes***	***Format***	***Nom***	***Description***
0	4	long	crown fuels	20 if no crown fuels, 21 if crown fuels exist (crown fuels = canopy height, canopy base height, canopy bulk density)
4	4	long	ground fuels	20 if no ground fuels, 21 if ground fuels exist (ground fuels = duff loading, coarse woody)
8	4	long	latitude	latitude (négative pour l'hémisphère sud)
12	8	double	loeast	offset to preserve coordinate precision (legacy from 16-bit OS days)
20	8	double	hieast	offset to preserve coordinate precision (legacy from 16-bit OS days)
28	8	double	lonorth	offset to preserve coordinate precision (legacy from 16-bit OS days)
36	8	double	hinorth	offset to preserve coordinate precision (legacy from 16-bit OS days)
44	4	long	loelev	élévation minimale
48	4	long	hielev	élévation maximale
52	4	long	numelev	nombre de classes d'élévation, -1 si > 100
56	400	long	elevation values	liste de valeurs d'élévation de type longs
456	4	long	loslope	pente minimale
460	4	long	hislope	pente maximale
464	4	long	numslope	nombre de classes de pente, -1 si > 100
468	400	long	slope values	liste de valeurs de pente de type longs

868	4	long	loaspect	orientation minimale
872	4	long	hiaspect	orientation maximale
876	4	long	numaspects	nombre de classes d'orientation, -1 si > 100
880	400	long	aspect values	liste de valeurs d'orientation de type longs
1280	4	long	lofuel	modèle de la valeur du fuel minimale
1284	4	long	hifuel	modèle de la valeur du fuel maximale
1288	4	long	numfuel	nombre de modèle de fuel -1 si > 100
1292	400	long	fuel values	liste de modèle de la valeur de fuel de type longs
1692	4	long	locover	couverture de la canopée minimale
1696	4	long	hicover	couverture de la canopée maximale
1700	4	long	numcover	nombre de classes de couverture de la canopée, -1 si > 100
1704	400	long	cover values	liste des valeurs de couverture de la canopée de type longs
2104	4	long	loheight	hauteur minimale de la canopée
2108	4	long	hiheight	hauteur maximale de la canopée
2112	4	long	numheight	nombre de classes de la hauteur de la canopée, -1 si > 100
2116	400	long	height values	liste de valeurs de la hauteur de la canopée de type longs
2516	4	long	lobase	hauteur minimale de la base de la canopée
2520	4	long	hibase	hauteur maximale de la base de la canopée
2524	4	long	numbase	nombre de classes de hauteur de la base de la canopée, -1 si > 100
2528	400	long	base values	**liste de valeurs de la hauteur de la base de la canopée** de type longs
2928	4	long	lodensity	minimum de la densité du couvert
2932	4	long	hidensity	maximum de la densité du couvert

2936	4	long	numdensity	nombre de classes de la densité du couvert, -1 si >100
2940	400	long	density values	liste de valeur de la densité du couvert de type longs
3340	4	long	loduff	minimum de poussière (duff)
3344	4	long	hiduff	maximum de poussière (duff)
3348	4	long	numduff	nombre de classes de poussière, -1 si > 100
3352	400	long	duff values	liste de valeur de poussière de type longs
3752	4	long	lowoody	ligneux grossiers minimal
3756	4	long	hiwoody	ligneux grossiers maximal
3760	4	long	numwoodies	nombre de classes de ligneux grossiers, -1 si > 100
3764	400	long	woody values	liste de valeurs de ligneux grossiers de type longs
4164	4	long	numeast	nombre de colonnes raster
4168	4	long	numnorth	nombre de ligne de raster
4172	8	double	EastUtm	X max
4180	8	double	WestUtm	X min
4188	8	double	NorthUtm	Y max
4196	8	double	SouthUtm	Y min
4204	4	long	GridUnits	unité linéaire : 0 = meters, 1 = feet, 2 = kilometers
4208	8	double	XResol	largeur de la taille de la cellule en GridUnits
4216	8	double	YResol	hauteur de la taille de la cellule en GridUnits
4224	2	short	EUnits	unité d'élévation : 0 = meters, 1 = feet
4226	2	short	SUnits	unité de la pente : 0 = degrees, 1 = percent
4228	2	short	AUnits	unité de l'orientation : 0 = Grass categories, 1 = Grass degrees, 2 = azimuth degrees
4230	2	short	FOptions	options du modèle de fuel : 0 = pas de

				modèle personnalisé ET pas de fichier de conversion, 1 = modèle personnalisé MAIS pas de fichier de conversion, 2 = pas de modèle personnalisé MAIS fichier de conversion, 3 = modèle de personnalisé ET fichier de conversion nécessaire
4232	2	short	CUnits	unité de couverture de la canopée : 0 = categories (0-4), 1 = percent
4234	2	short	HUnits	unité de hauteur de canopée : 1 = meters, 2 = feet, 3 = m x 10, 4 = ft x 10
4236	2	short	BUnits	unité de hauteur de la base de la canopée : 1 = meters, 2 = feet, 3 = m x 10, 4 = ft x 10
4238	2	short	PUnits	unité de la densité du couvert : 1 = kg/m³, 2 = lb/ft³, 3 = kg/m³ x 100, 4 = lb/ft³ x 1000
4240	2	short	DUnits	unité de la poussière : 1 = Mg/ha x 10, 2 = t/ac x 10
4242	2	short	WOptions	options du ligneux grossiers (1 si la bande ligneux grossier est présent)
4244	256	char[]	ElevFile	nom du fichier d'élévation
4500	256	char[]	SlopeFile	nom du fichier de pente
4756	256	char[]	AspectFile	nom du fichier de l'orientation
5012	256	char[]	FuelFile	nom du fichier du modèle de fuel
5268	256	char[]	CoverFile	nom du fichier de la couverture de la canopée
5524	256	char[]	HeightFile	nom du fichier de la hauteur de la canopée
5780	256	char[]	BaseFile	nom du fichier de la base de la canopée
6036	256	char[]	DensityFile	nom du fichier de la densité du couvert
6292	256	char[]	DuffFile	nom du fichier de poussière
6548	256	char[]	WoodyFile	nom du fichier des ligneux grossiers
6804	512	char[]	Description	fichier de description LCP

Chris Toney, 2009-02-14

Chapitre XLVII Leveller --- Daylon Leveller Heightfield

(http://www.lizardtech.com/)

Leveller heightfields stocke des valeurs d'élévation sur 32 bit. Les formats versions 4 à 7 sont géré avec différents canevas (voir ci-dessous). L'extension du fichier pour Leveller heightfields est « TER » (qui est le même que celui de Terragen, mais le pilote ne reconnais que les fichiers Leveller).

Les bloques sont organisé sous forme de ligne scannée à haut pixel (lignes) avec la première ligne scannée en haut (nord) du bord du DEM, et les pixels adjacents sur chaque ligne augmentant de la gauche vers la droite (ouest vers l'est).

Le type de bande est toujours Float32, même avec les formats versions 4 et 5 utilise physiquement des points fixes de 16.16. Le pilote les convertit en point flottant.

A Lecture

`dataset::GetProjectionRef()` renverra seulement un système de coordonnée pour les fichiers de versions 4 à 6.

`dataset::GetGeoTransform()` renverra une simple transformation mondiale avec une origine centrée pour les formats 4 à 6. Pour la version 7, il renvoie une transformation mondiale réelle sauf pour les rotations. La transformation de l'identité n'est pas considéré comme une condition d'erreur ; Les documents de Leveller les utilisent souvent.

`band::GetUnitType()` rapportera les unités de mesure utilisé par le fichier au lieu de convertir les types inhabituels en mètre. Une liste de type d'unité se trouve dans le module *levellerdataset.cpp*.

`band::GetScale()` et `band::GetOffset()` renverra la transformation physique vers celle logique (c'est à dire brute vers celle du monde réelle) pour les données d'élévation.

B Écriture

L'appel `dataset::Create()` est géré, mais pour les fichiers de version 7 seulement.

`band::SetUnitType()` peut être définie à n'importe quel type d'unité listé dans le module levellerdataset.cpp.

`dataset::SetGeoTransform()` ne doit pas inclure les données de rotation.

De même que le pilote Terragen, l'option `MINUSERPIXELVALUE` doit être définie. Cela laisse le pilote cartographier correctement les élévations logiques (le monde réel) vers les élévations physiques.

Les informations d'en-tête sont écrit lors du premier appel à *band::IWriteBlock*.

C Historique

- v1.2 (Jul 17/07) : ajout de la version 7 et gestion de *Create*.
- v1.1 (Oct 20/05) : correction des erreurs dans coordsys et dans « elev scaling » errors.

Lire également :

- Implémenté dans gdal/frmts/leveller/levellerdataset.cpp.
- Le SDK de Leveller, qui documente le format Leveller : http://www.daylongraphics.com/products/leveller/dev/index.htm

Chapitre XLVIII MEM -- In Memory Raster

GDAL gère la possibilité de prendre en charge des rasters dans un format temporaire en mémoire. Cela est d'abord utile pour les jeux de données temporaires dans les scripts ou en interne dans des applications. Il n'est généralement d'aucune utilité à l'utilisateur final. Les jeux de données en mémoire doivent gérer la plupart des espèces d'informations auxiliaires dont les méta-données, les systèmes de coordonnées, le géoréférencement, les points d'amer, l'interprétation des couleurs, les tables de couleur et tous les types de données des pixel.

A Format des noms des jeux de données

Il est possible d'ouvrir un tableau existant en mémoire. Pour cela, construire un nom de jeux de données de la forme suivante :

```
MEM:::option=value[,option=value...]
```

Par exemple :

```
MEM:::DATAPOINTER=342343408,PIXELS=100,LINES=100,BANDS=3,DATATYPE=Byte
,
   PIXELOFFSET=3,LINEOFFSET=300,BANDOFFSET=1
```

- `DATAPOINTER` : pointeur vers le premier pixel de la première bande représentée par un entier long. Note : cela peut ne pas fonctionner sur les plate-formes où un entier long est en 32 bites et un pointeur est en 64 bites. (obligatoire)
- `PIXELS` : largeur d'un raster en pixel (obligatoire)
- `LINES`: hauteur d'un raster en ligne (obligatoire)
- `BANDS` : nombre de bande, défaut à 1 (optionnel)
- `DATATYPE` : nom du type de données, comme retourné par `GDALGetDataTypeName()` (par exemple Byte, Int16), Byte par défaut. (optionnel)
- `PIXELOFFSET` : distance en bytes entre le début d'un pixel et le suivant sur la même ligne scannée. (optionnel)
- `LINEOFFSET` : distance en bytes entre le début d'une ligne scannée et la suivante (optionnel)

- `BANDOFFSET` : distance en bytes entre le début d'une bande de données et la suivante.

B Options de création

- Il n'y a aucune options de création de gérées.
- Le format MEM est un des seuls qui gère la méthode `AddBand()`. La méthode `AddBand()` gère les options DATAPOINTER, PIXELOFFSET et LINEOFFSET pour faire référence à un tableau existant en mémoire.

Chapitre XLIX MFF2 -- Vexcel MFF2 Image

GDAL gère le format de fichier raster Image MFF2 en lecture, mise à jour et création. Le format MFF2 (Multi-File Format 2) a été définie pour s'ajuster dans les bases de données Vexcel Hierarchical Key-Value (HKV), qui peut stocker des données binaires ainsi que des paramètres ASCII. Ce format est d'abord utilisé en interne dans le système de traitement Vexcel Insar.

Pour sélectionner un jeu de données MFF2, sélectionnez le répertoire contenant les fichiers *attrib*, et *image_data* du jeu de données.

Pour l'instant seulement la latitude/longitude et la projection UTM sont gérés (georef.projection.name = ll ou georef.projection.name = utm), par transformation affine calculée à partir des points d'amer en lat/long. Pour n'importe quelle action, si les points d'amer sont disponibles dans un fichier de géoreférencement , ceux-ci sont renvoyés avec le jeu de données. Les fichiers nouvellement crées (avec un type de MFF2) sont toujours des raster brutes sans information de géoréférencement. pour la lecture et la création, tous les types de données (réel, entier, et complexe avec une profondeur en bit de 8, 16, 32) devraient être gérés.

Warning!

Lors de la création d'un nouveau fichier MFF2, assurez-vous de définir la projection avant de définir la transformation spatiale (cela est nécessaire parce que la transformation spatiales est stockée en interne sous forme de 5 points d'amer au sol en latitude/longitude et la projection est nécessaire à la conversion).

Note!

implémenté dans gdal/frmts/raw/hkvdataset.cpp.

A Détails du format

1 Structure générale du MFF2

Un « fichier » MFF2 est en réalité un ensemble de fichiers stockés dans un répertoire contenant un fichier d'en-tête ASCII nommé « attrib » et des données d'image binaire nommé « image_data ».En option, il peut y avoir un fichier « georef » ASCII contenant

des informations de projection et de géoréférencement, et un fichier « image_data_ovr » (pour les données image binaire « image_data ») contenant les aperçus tuilés de l'image au format TIFF. Les fichiers ASCII sont arrangés sous forme de pairs clé=valeur. Les pairs autorisées pour chaque fichier sont décrites ci-dessous.

2 Le fichier "attrib"

Au minimum, le fichier « attrib » doit contenir l'étendue de l'image, la taille en pixel en octets, l'encodage des pixels et le type des données, et l'ordre des octets. Par exemple :

```
extent.cols    = 800
extent.rows    = 1040
pixel.size     = 32
pixel.encoding = { unsigned twos_complement *ieee_754 }
pixel.field    = { *real complex }
pixel.order    = { lsbf *msbf }
version        = 1.1
```

définie une image de 1040 lignes et 800 pixels pour l'étendue. Les pixels sont des entiers de 32 bits dont l'ordre est « l'octet le plus significatif en premier » (msbf), encodé selon la spécification ieee_754. Dans un MFF2, lorsqu'une valeur appartient à un certain sous-ensemble (par exemple pixel.order doit être soit lsbf ou msbf) toutes les options sont affichées entre crochets, et celui approprié pour le fichier en cours est indiqué par une « * ».

Le fichier peut aussi contenir les lignes suivantes indiquant le nombre de canaux de données, et comment ils sont entrelacés à l'intérieur du fichier de données binaire.

```
channel.enumeration = 1
channel.interleave = { *pixel tile sequential }
```

3 Le fichier "image_data"

Le fichier « image_data » est constitué de données binaire brute avec une étendue, l'encodage du pixel, et le nombre de canaux comme indiqué dans le fichier « attrib ».

4 Le fichier "georef"

Le fichier « georef » est utilisé pour décrire des informations de géocodage et de projection pour les données binaires. Par exemple,

```
top_left.latitude            = 32.93333333333334
top_left.longitude           = 130.0
top_right.latitude           = 32.93333333333334
top_right.longitude          = 130.5
bottom_left.latitude         = 32.50000000000001
bottom_left.longitude        = 130.0
bottom_right.latitude        = 32.50000000000001
bottom_right.longitude       = 130.5
centre.latitude              = 32.71666666666668
centre.longitude             = 130.25
projection.origin_longitude  = 0
```

```
projection.name              = ll
spheroid.name                = wgs-84*
```

décrit un image projetée en latitude/longitude (ll) orthogonale, avec les latitudes et longitudes basé sur l'ellipsoïde wgs-84. Depuis la version 1.1 de MFF2, top_left renvoie au coin en haut à gauche du pixel en haut à gauche. top_right renvoie au coin en haut à droite du pixel en haut à droite. bottom_left renvoie au coin en bas à gauche du pixel en bas à gauche. bottom_right renvoie au coin en bas à droite du pixel en bas à droite. centre renvoie au centre des quatre coins définie plus haut (le centre de l'image).

Mathématiquement, pour une image Npix par Nline, les coins et le centre sous forme de coordonnées (pixel,line) pour la version 1.1 de MFF2 sont :

```
top_left: (0,0)
top_right: (Npix,0)
bottom_left: (0,Nline)
bottom_right: (Npix,Nline)
centre: (Npix/2.0,Nline/2.0)
```

Ces calculs sont réalisé en virgule flottante (c'est à dire que les coordonnées du centre peuvent prendre des valeurs non entières).

Remarquez que les coins sont toujours exprimé en latitude/longitude, même pour les images projetées.

5 Projection gérées

image projetée latitude/longitude Orthogonal -ll, avec les latitudes parallèles aux lignes et les longitudes parallèles aux colonnes. Paramètres : nom du sphéroïde, projection, longitude d'origine (longitude à l'origine des coordonnées de la projection). Si non définie, la valeur est par défaut la longitude central de l'image en sortie basée sur ces limites de projection.

Image projeté en Universal Transverse Mercator – utm. Paramètres : nom du sphéroïde, projection origine de la projection (méridien central pour la projection utm). Le méridien centrale doit être le méridien au centre de la zone UTM, par exemple 3 degré, 9 degré, 12 degré, etc. Si cela n'est pas définie ou définie à un méridien central UTM valide, le lecteur doit annuler la valeur au méridien central valide le plus proche basé sur la longitude centrale de l'image en sortie. La latitude à l'origine, de la projection UTM est toujours de 0 degré.

6 Ellipsoïdes reconnus

Le format MFF2 associe les noms suivants avec les paramètres du rayon à l'équateur et le coefficient à l'aplatissement de l'ellipsoïde :

```
airy-18304:            6377563.396      299.3249646
modified-airy4:        6377340.189      299.3249646
australian-national4:  6378160          298.25
bessel-1841-namibia4:  6377483.865      299.1528128
bessel-18414:          6377397.155      299.1528128
clarke-18584:          6378294.0        294.297
clarke-18664:          6378206.4        294.9786982
```

```
clarke-18804:          6378249.145      293.465
everest-india-18304:   6377276.345      300.8017
everest-sabah-sarawak4:6377298.556      300.8017
everest-india-19564:   6377301.243      300.8017
everest-malaysia-19694:6377295.664      300.8017
everest-malay-sing4:   6377304.063      300.8017
everest-pakistan4:     6377309.613      300.8017
modified-fisher-19604: 6378155          298.3
helmert-19064:         6378200          298.3
hough-19604:           6378270          297
hughes4:               6378273.0        298.279
indonesian-1974:       6378160          298.247
international-1924:    6378388          297
iugc-67:               6378160.0        298.254
iugc-75:               6378140.0        298.25298
krassovsky-1940:       6378245          298.3
kaula:                 6378165.0        292.308
grs-80:                6378137          298.257222101
south-american-1969:   6378160          298.25
wgs-72:                6378135          298.26
wgs-84:                6378137          298.257223563
ev-wgs-84:             6378137          298.252841
ev-bessel:             6377397          299.1976073
```

7 Explication des champs

- **channel.enumeration :** (optionnelle seulement nécessaire pour les multibandes) Nombre de canaux dedonnées (par exemple 3 pour rgb)
- **channel.interleave = { *pixel tile sequential } :** (optionnelle seulement nécessaire pour les multibandes) Pour des données multibandes, indique comment les canaux sont entrelacés. *pixel indique des les données sont stockés en valeur de rouge, vert, bleu, rouge, vert, bleu etc. par opposition à (ligne de valeurs rouge) (ligne de valeur de vert) (ligne de valeur de bleu) ou (canal complet de rouge) (canal complet de vert) (canal complet de bleu)
- **extent.cols :** nombre de colonne de données.
- **extent.rows :** nombre de ligne de données.
- **pixel.encoding = { *unsigned twos-complement ieee-754 } :** combiné avec pixel.size et pixel.field pour donner le type de données :

```
(encoding, field, size)- type
(unsigned, real, 8)- unsigned byte data
(unsigned, real, 16)- unsigned int 16 data
(unsigned, real, 32)- unsigned int 32 data
(twos-complement, real, 16)- signed int 16 data
(twos-complement, real, 32)- signed int 32 data
(twos-complement, complex, 64)- complex signed int 32 data
(ieee-754, real, 32)- real 32 bit floating point data
(ieee-754, real, 64)- real 64 bit floating point data
(ieee-754, complex, 64)- complex 32 bit floating point data
(ieee-754, complex, 128)- complex 64 bit floating point data
```

- **pixel.size :** taille d'un pixel d'un canal (bits).
- **pixel.field = { *real complex } :** si la données est réelle ou complexe.
- **pixel.order = { *lsbf msbf } :** ordonnancement des bytes des données (least ou

most significant byte first).

- **version :** (seulement dans les versions récentes – si non présent, une version plus vielle est présumé) Version de mff2.

Chapitre L MrSID --- Multi-resolution Seamless Image Database

MrSID est une technologie de compression d'image basée sur la transformée par ondelette qui peut être utilisé à la fois avec et sans perte. Cette technologie a été acquise sous sa forme originelle par les Laboratoires Nationaux de Los Alamos (LANL), où il a été développé sous l'égide du gouvernement U.S. pour le stockage des empreintes digitales pour le FBI. Maintenant elle est développée et distribuée par la société LizardTech.

Ce pilote gère la lecture des fichiers images MrSID en utilisant le kit de développement de logiciel d'encodage de LizardTech (DSDK). Ce DSDK n'est pas un logiciel libre, vous devez contacter LizardTech pour l'obtenir (voyez le lien à la fin de cette section). Si vous utilisez GCC, s'il vous plaît, assurez-vous que vous avez le même compilateur qui a été utilisé pour la compilation de DSDK. C'est une bibliothèque en C++, vous pouvez donc avoir des incompatibilités dans le name mangling entre les différentes version de GCC (2.95.x et 3.x).

Les dernières versions de DSDK gère également le décodage du format de fichier JPEG2000, ce pilote peut aussi être utilisé pour le format JPEG2000.

A Méta-données

Les méta-données MrSID sont traduites de manière transparente dans les chaînes de méta-données de GDAL. Les fichiers au format MrSID contiennent un ensemble de balises de méta-données standards telles que : IMAGE__WIDTH (contient la largeur de l'image), IMAGE__HEIGHT (contient la hauteur de l'image), IMAGE__XY_ORIGIN (contient les coordonnées x et y de l'origine), IMAGE__INPUT_NAME (contient le nom ou les noms des fichiers utilisés pour créer l'image MrSID) etc. Les clés des méta-données de GDAL ne peuvent pas contenir certains caractères ':' et '=', mais les balises MrSID standards contiennent toujours le symbole ':' dans les noms des balises. Ces caractères sont remplacés dans GDAL par '_' pendant la traduction. Si vous utilisez d'autres logiciels pour travailler sur les images MrSID attendez vous à ce que les noms des clés des méta-données soient affichés différemment.

B Géoréférencement

Les images MrSID peuvent contenir des informations de géoréférencement et du système

de coordonnées sous la forme de clé géographique (GeoKeys) GeoTiff, traduites dans les enregistrements des méta-données. Toutes ces GeoKeys sont extraites proprement et utilisées par le pilote. Malheureusement, il y a une contrainte : les anciens encodeurs MrSID avaient un bug qui entraînait des GeoKeys incorrectes, stocké dans les fichiers MrSID. Ce bug a été corrigé dans la version 1.5 du logiciel MrSID , mais si vous avez un vieil encodeur ou des fichiers crées avec un vieil encodeur, vous ne pouvez pas utiliser leurs informations de géoréférencement.

C Options de création

L'écriture au format MrSID est seulement géré si GDAL est compilé avec l'ESDK 5.x ou supérieur de MrSID. Celui-ci est normalement vendu uniquement par Lizardtech d'une manière contrôlée (bien que le DSDK soit libre/gratuit (?) pour quiconque l'utilise dans les contraintes de l'accord de licence). Si vous avez l'EDSK, il peut être utilisé pour écrire des fichiers MrSID. Les options de création suivante sont gérées :

- **WORLDFILE :** Yes pour écrire le fichier world d'ESRI (avec l'extension .sdw).
- **VERSION :** peut être 2 pour une version 2 des fichiers MrSID ou 3 pour une version 3 des fichiers MrSID.
- **COMPRESSION :** Utilisé seulement pour la version 2 des fichiers MrSID. Indique le taux de compression désiré. La génération 2 du format MrSID ne peut pas compresser une image sans perte ; donc zéro n'indique pas une compression numérique sans perte. Une valeur de 1 peut être utilisé pour une plus grande fidélité, mais une valeur de 20 produira de meilleur résultat (une compression visuellement sans perte). Vingt signifie un taux de compression de 20:1 (l'image sera compressée au 1/20 de sa taille originelle).
- **FILESIZE :** Utilisé seulement pour la version 3 du format MrSID. Indique la taille en octet du fichier en sortie. Utiliser « 0 » pour une compression sans perte.
- **TWOPASS :** Utilisé seulement avec les fichiers MrSID version 3. Indique qu'un algorithme d'optimisation à deux passages doit être utilisé pendant la compression.

Lisez également :

- Implementé dans *gdal/frmts/mrsid/mrsiddataset.cpp*.
- LizardTech's Web site

Chapitre LI Compression Lidar de MrSID/MG4 / Fichiers view de Cloud ponctuel

Ce pilote fournie un moyen de visualiser un fichier LiDAR compressé en MrSID/MG4 comme un MNT raster. Les spécificités de la conversion dépend de la taille de la cellule désirée, des critères de filtre, des méthodes d'agrégation et éventuellement d'autres paramètres. Pour cette raison, **le meilleur moyen de lire un fichier LiDAR compressé en MrSID/MG4 est en le référençant dans un fichier View (.view), qui paramétrise sa conversion en raster. Le pilote lira un fichier MG4 directement, cependant il utilise des paramètres de rasterisation par défaut qui peuvent ne pas produire un rendu correcte**. Le contenu d'un fichier View est décrit dans la spécification documents de visualisation de MrSID/MG4 LiDAR.

MrSID/MG4 est une technologie de compression de cloud ponctuel à base d'ondelettes. Vous pouvez y penser comme un fichier LAS, seulement plus petit et avec un index spatial interne. Il a été développé et est distribué par LizardTech. Ce pilote gère la lecture de fichier LiDAR MG4 en utilisant le kit de développement de décodage (DSDK). **Ce DSDK est gratuitement distribué ; mais il n'est pas un logiciel open source. Vous devez contacter LizardTech pour l'obtenir (voyez le lien en bas de ce chapitre).**

A Exemple de fichier View (à partir de la spécification des documents View)

1 Fichier .view le plus simple possible

Le plus simple moyen pour afficher un fichier MG4 est de l'envelopper dans un fichier View (.view) comme cela. Ici, la référence relative au fichier MG4 signifie que le fichier doit exister dans le même répertoire que le fichier .view. Puisque nous ne mappons pas les bandes de façon explicite, nous obtenons la valeur par défaut, qui est uniquement l'élévation. Par défaut, nous agrégeons basé sur la moyenne. Autrement dit, si deux points (ou plus) land sur une seule cellule, nous allons exposer la moyenne des deux. Il n'y a aucun filtrage ici, donc nous aurons tous les points indépendamment du code de classification ou du numéro de retour. Puisque le type de données natif d'élévation est

"Float64", qui est le type de données de la bande que nous allons exposer.

```
<PointCloudView>
    <InputFile>Tetons.sid</InputFile>
</PointCloudView>
```

2 Découpe des données

Ceci est similaire à l'exemple ci-dessus, mais nous utilisons la balise optionnelle ClipBox pour sélectionner une étendue de 300 mètres nord-sud via le cloud. Si nous voulions découper dans les directions Est-Ouest, nous aurions pu spécifier ceci explicitement au lieu d'utiliser NOFITLER pour ceux-ci. De même, nous pourrions aussi découper dans la direction Z.

```
<PointCloudView>
    <InputFile>Tetons.sid</InputFile>
    <ClipBox>505500 505800 NOFILTER NOFILTER</ClipBox>
</PointCloudView>
```

3 Expose as a bare earth (Max) DEM

Ici, nous exposons une seule bande (élévation) mais désirons seulement les points qui ont été classifié comme "Ground". *ClassificationFitler* définie une valeur de 2 - Le code ASPRS Point Class qui stipule les points "Ground". De plus, au lieu de la méthode d'agrégation "Moyen" par défaut, nous spécifiions "Max". Cela signifie que si deux points (ou plus) se situe dans une seule cellule, nous exposons la plus grande des deux valeurs d'élévation.

```
<PointCloudView>
    <InputFile>E:\ESRIDevSummit2010\Tetons.sid</InputFile>
    <Band> <!-- Max Bare Earth-->
        <Channel>Z</Channel>
        <AggregationMethod>Max</AggregationMethod>
        <ClassificationFilter>2</ClassificationFilter>
    </Band>
</PointCloudView>
```

4 Image d'intensité

Ici nous exposons une image d'intensité à partir du cloud ponctuel.

```
<PointCloudView>
    <InputFile>Tetons.sid</InputFile>
    <Band>
        <!-- All intensities -->
        <Channel>Intensity</Channel>
    </Band>
</PointCloudView>
```

5 Images RVB

Certaines images cloud ponctuelle inclut des données RVB. Si c'est le cas, vous pouvez utiliser un fichier .view comme celui-ci pour exposer ces données.

```
<PointCloudView>
    <InputFile>Grass Lake Small.xyzRGB.sid</InputFile>
    <Band>
        <Channel>Red</Channel>
    </Band>
    <Band>
        <Channel>Green</Channel>
    </Band>
    <Band>
        <Channel>Blue</Channel>
    </Band>
</PointCloudView>
```

B Gestion de l'écriture

Ce pilote ne gère pas l'écriture de fichiers MG4.

C Limitations de l'implémentation actuelle

Seulement une balise *<InputFile>* est gérée. Elle doit référencer un fichier MG4.

La seule *<InterpolationMethod>* qui est gérée est *<None>* (défaut). Utilisez cela pour définir une valeur NODATA si la valeur par défaut (valeur maximale de ce type de données) n'est pas ce que vous souhaitez. Voyez la spécification "document View" pour plus de détails.

Il y a une insuffisance dans la vérification des erreurs pour les erreurs de format et les paramètres invalides. Plusieurs entrées invalides échoueront probablement silencieusement.

D Voir également

- Implémenté dans *gdal/frmts/mrsid_lidar/gdal_MG4Lidar.cpp*
- Spécification des documents de visualisation de MrSID/MG4 LiDAR
- Site web de LizardTech

Chapitre LII MSG -- Météosat Seconde Génération

Ce pilote implémente la gestion de la lecture pour les fichiers Meteosat Second Generation. Le nom de ces fichiers sont du type *H-000-MSG1__-MSG1_____-HRV_____-000007___-200405311115-C_,* distribué habituellement dans une structure arborescente avec des dates (par exemple 2004/05/31 pour le fichier mentionné).

Les fichiers MSG sont compressé par ondelette. Une bibliothèque sous licence Eumetsat est nécessaire (« Wavelet Transform Software » - http://www.eumetsat.int/en/dps/helpdesk/tools.html#wavelet). Le logiciel est compatible sous les systèmes Microsoft Windows, Linux et Solaris, et il fonctionne sous 32 bits et 64 bits ainsi que les architectures mélangées. Elle est sous licence logiciel et est disponible après acceptation de la licence logiciel WaveLet Transform lors d'une procédure électronique.

Une partie des sources du fichier *xrithdr_extr.c* provenant XRIT2PIC est utilisé pour parser les en-têtes MSG. Ces sources sont sous licence GNU General Public Licence publié par la Fondation Free Software.

Ce pilote n'est pas activé par défaut. Lisez la partie « instructions de compilation » pour la manière d'inclure ce pilote dans votre bibliothèque GDAL.

A Instructions de compilation

Téléchargez la bibliothèque Eumestat pour la décompression par ondelette. C'est un fichier nommé PublicDecompWt.zip. Décompressez son contenu dans un sous-répertoire avec le même nom (dans frmts/msg).

Si vous compilez avec Visual Studio 6.0, décompressez le makefiles .vc pour *PublicDecompWT* à partir du fichier *PublicDecompWTMakefiles.zip*.

Si vous compilez en utilisant GNUMakefile, utilisez l'option `--with-msg` pour activer le pilote MSG :

```
./configure --with-msg
```

S'il se trouve que certain ajustement soit nécessaire dans le makefile et/ou les fichiers

sources, s'il vous plaît commitez" les. La bibliothèque Eumetsat promet d'être indépendant de la plateforme, mais puisque nous travaillons sous Microsfot Windows et Visual Studio 6.0, nous n'avons pas la possibilité de vérifier si l'ensemble du pilote msg l'est. De plus, appliquez les étapes 4 à 7 du tutorial d'Implémentation du pilote de GDAL, section "Ajouter un pilote à l'arbre de GDAL".

Le wiki de MSG est disponible sur http://trac.osgeo.org/gdal/wiki/MSG. Il est dédié à la documentation de la compilation et astuce d'usage.

B Spécification des sources des jeux de données

Il est possible de sélectionner des fichiers individuels pour l'ouverture. Dans ce cas, le pilote collectera les fichiers qui correspondent aux autres bandes de la même image, et composera correctement l'image.

Exemple avec gdal_translate.exe :

```
gdal_translate C:\hrit_a\2004\05\31\H-000-MSG1__-MSG1________-
HRV_____-000008___-200405311115-C_ c:\output\myimage.tif
```

Il est également possible d'utiliser la syntaxe suivantes pour l'ouverture des fichiers MSG :

```
MSG(source_folder,timestamp,
(canal,canal,...,canal),use_root_folder,data_conversion,nr_cycles,step
)
```

- `source_folder` : un chemin vers la structure arborescente qui contient les fichiers.
- `timestamp` : 12 digits représentant une date/heure qui identifie les 114 fichiers des 12 images de cette heure, par exemple 200501181200
- `canal` : un nombre compris entre 1 et 12, représentant chacun des 12 canaux disponibles. Lorsqu'un seul canal est disponible, les crochets sont optionnels.

use_root_folder : Y pour indiquer que les fichiers résident directement à la racine du répertoire source_folder définie. N pour indiquer que les fichiers résident dans les répertoires structurés en date : source_folder/YYYY/MM/DD

- `data_conversion` :
  - `N` pour garder les valeurs DN originelles de 10 bits. Le résultat est UInt16.
  - `B` pour convertir en 8 bits (commode pour les images GIF et JPEG). Le résultat est en Byte.
  - `R` pour réaliser des calibration radio-métrique et obtenir le résultat en mW/m2/sr/(cm-1)-1. Le résultat est Float32.
  - `L` pour réaliser des calibrations radio-métrique et obtenir les résultat en W/m2/sr/um. Le résultat est Float32.
  - `T` pour obtenir la réflectivité pour les bandes visible (1, 2, 3 et 12) et la température en degré Kelvin pour les bandes infrarouges (toutes les autres bandes). Le résultat est Float32.
- `nr_cycles` : un nombre qui indique le nombre de cycles consécutifs pour être inclus dans le même fichier (séries temporelle). Ceux-ci sont ajouté aux bandes additionnelles.
- `step` : un nombre qui indique quelle est la taille de l'étape quand des cycles multiples sont choisit. Par exemple : toutes les 15 minutes : step = 1, toutes les

30 minutes : step = 2 etc. Notez que les cycles sont des multiple de 15, vous ne pouvez donc pas avoir des images entre ces moments (step est un entier).

Exemples avec gdal_translate.exe :

Exemple d'appel pour récupérer une image MSG 200501181200 avec des bandes 1, 2 et 3 au format IMG :

```
gdal_translate -of HFA MSG(\\pc2133-24002\RawData\,200501181200,
(1,2,3),N,N,1,1) d:\output\outfile.img
```

Au format JPG, et convertir des images e 10 bites en image 8 bits en divisant toutes les valeurs par 4 :

```
gdal_translate -of JPEG MSG(\\pc2133-24002\RawData\,200501181200,
(1,2,3),N,B,1,1) d:\output\outfile.jpg
```

Même chose, mais en réordonnant les bandes dans l'image JPEG pour ressembler à une image RVB :

```
gdal_translate -of JPEG MSG(\\pc2133-24002\RawData\,200501181200,
(3,2,1),N,B,1,1) d:\output\outfile.jpg
```

Sortie Geotiff, seule la bande 2, valeurs originales en 10 bits :

```
gdal_translate -of GTiff MSG(\\pc2133-
24002\RawData\,200501181200,2,N,N,1,1) d:\output\outfile.tif
```

Bande 12 :

```
gdal_translate -of GTiff MSG(\\pc2133-
24002\RawData\,200501181200,12,N,N,1,1) d:\output\outfile.tif
```

La même bande 12 avec une calibration radiométrique en mW/m2/sr/(cm-1)-1 :

```
gdal_translate -of GTiff MSG(\\pc2133-
24002\RawData\,200501181200,12,N,R,1,1) d:\output\outfile.tif
```

Récupérer les données de c:hrit-data20050118 au lieu de \pc2133-24002RawData... :

```
gdal_translate -of GTiff MSG(c:\hrit-
data\2005\01\18,200501181200,12,Y,R,1,1) d:\output\outfile.tif
```

Autre option pour faire la même chose (notez la différence dans le Y et le N pour l'utilisation du paramètre “use_root_folder” :

```
gdal_translate -of GTiff MSG(c:\hrit-data\,200501181200,12,N,R,1,1)
d:\output\outfile.tif
```

Sans calibration radiométrique, mais pour 10 cycles consécutifs (donc de 1200 à 1415) :

```
gdal_translate -of GTiff MSG(c:\hrit-data\,200501181200,12,N,N,10,1)
```

```
d:\output\outfile.tif
```

10 cycles,mais toutes les heures (donc de 1200 à 2100) :

```
gdal_translate -of GTiff MSG(c:\hrit-data\,200501181200,12,N,N,10,4)
d:\output\outfile.tif
```

10 cycles, toutes les heures, et les bandes 3, 2 et 1 :

```
gdal_translate -of GTiff MSG(c:\hrit-data\,200501181200,
(3,2,1),N,N,10,4) d:\output\outfile.tif
```

C Géoréférencement et projection

Les images utilisent la projection de vue satellites géo-stationnaires. La plupart des logiciels SIG ne reconnaissent pas cette projection (nous savons seulement que le logiciel ILWIS possède cette projection), mais `gdalwarp.exe` peut être utilisé pour reprojeter les images.

Lisez également :

- Implémenté dans *gdal/frmts/msg/msgdataset.cpp*.
- http://www.eumetsat.int - Organisation Européenne pour l'Exploitation des Satellites Météorologique

Chapitre LIII MSGN -- Meteosat Second Generation (MSG) Format Natif d'Archive (.nat)

GDAL gère la lecture seulement des fichiers natifs MSG. Ces fichiers peuvent avoir entre 1 et 12 bandes, toutes à une résolutions de 10 bits.

La gestion pour la 12e bande (HRV - High Resolution Visible) incluse. Cela est implémenté dans un sous jeu, c'est à dire, il est nécessaire de préfixer le nom du fichier avec la balise "HRV:".

De même, il est possible d'obtenir des valeurs de radiance de virgule flottante à la place des nombres numériques de 10 bit (DNs). Ce sous jeu est accédé en préfixant le nom du fichier avec la balise "RAD:".

Le géoréférencement est actuellement géré mais les résultats ne sont pas acceptable (pas assez précis) en fonction de vos besoins. L'astuce actuelle est d'implémenter la projection géostationnaire CGMS directement, en utilisant le code disponible sur EUMETSAT.

Chapitre LIV NetCDF : Network Common Data Form

Ce format est géré en lecture et écriture. NetCDF est une interface pour l'accès au données orienté 'tableau' et est utilisé pour représenter des données scientifique.

Les méta-données des valeurs rempli ou la compatibilité arrière des valeurs manquantes sont préservés comme valeur NODATA lorsque cela est disponible.

Note : Implémenté dans *gdal/frmts/netcdf/netcdfdataset.cpp*.

A Multiple Image Handling (sous-jeu de données)

Nework Command Data Form est un conteneur pour plusieurs tableaux différents la plupart du temps utilisé pour stocker des jeux de données scientifiques. Un fichier netCDF peut contenir plusieurs jeux de données. Ils peuvent différer par la taille, le nombre de dimensions et peuvent représenter des données pour différentes régions.

Si le fichier contient seulement un tableau netCDF et que celle-ci est une image, elle peut être accédé directement mais si le fichier contient de multiples images il peut être nécessaire d'importer le fichier via un processus en deux étapes.

La première étape est d'obtenir un rapport des images qui composent le jeu de données dans le fichier en utilisant `gdalinfo`, puis d'importer les images désirées en utilisant `gdal_translate`. La commande `gdalinfo` liste tous les sous-jeu de données multidimensionnel à partir du fichier netCDF d'entrée.

Le nom des images individuelles sont assigné à l'entrée de méta-données *SUBDATASET_n_NAME*. La description pour chaque image se trouve dans l'entrée de méta-données *SUBDATASET_n_DESC*. Pour netCDF les images suivront ce format : *NETCDF:filename:variable_name*

où *filename* esy le nom du fichier en entrée, et *variable_name* est le jeu de données sélectionné dans le fichier.

Pour la seconde étape vous fournissez ce nom pour `gdalinfo` pour obtenir les informations sur le jeu de données ou `gdal_translate` pour lire le jeu de données.

Par exemple, nous voulons lire les données d'un fichier NetCDF :

```
$ gdalinfo sst.nc
Driver: netCDF/Network Common Data Format
Size is 512, 512
Coordinate System is `'
Metadata:
    NC_GLOBAL#title=IPSL  model output prepared for IPCC Fourth
Assessment SRES A2 experiment
    NC_GLOBAL#institution=IPSL (Institut Pierre Simon Laplace, Paris,
France)
    NC_GLOBAL#source=IPSL-CM4_v1 (2003) : atmosphere : LMDZ (IPSL-
CM4_IPCC, 96x71x19) ; ocean ORCA2 (ipsl_cm4_v1_8, 2x2L31); sea ice LIM
(ipsl_cm4_v
    NC_GLOBAL#contact=Sebastien Denvil,
sebastien.denvil@ipsl.jussieu.fr
    NC_GLOBAL#project_id=IPCC Fourth Assessment
    NC_GLOBAL#table_id=Table O1 (13 November 2004)
    NC_GLOBAL#experiment_id=SRES A2 experiment
    NC_GLOBAL#realization=1
    NC_GLOBAL#cmor_version=9.600000e-01
    NC_GLOBAL#Conventions=CF-1.0
    NC_GLOBAL#history=YYYY/MM/JJ: data generated; YYYY/MM/JJ+1 data
transformed  At 16:37:23 on 01/11/2005, CMOR rewrote data to comply
with CF standards and IPCC Fourth Assessment requirements
    NC_GLOBAL#references=Dufresne et al, Journal of Climate, 2015, vol
XX, p 136
    NC_GLOBAL#comment=Test drive
Subdatasets:
    SUBDATASET_1_NAME=NETCDF:"sst.nc":lon_bnds
    SUBDATASET_1_DESC=[180x2] lon_bnds (64-bit floating-point)
    SUBDATASET_2_NAME=NETCDF:"sst.nc":lat_bnds
    SUBDATASET_2_DESC=[170x2] lat_bnds (64-bit floating-point)
    SUBDATASET_3_NAME=NETCDF:"sst.nc":time_bnds
    SUBDATASET_3_DESC=[24x2] time_bnds (64-bit floating-point)
    SUBDATASET_4_NAME=NETCDF:"sst.nc":tos
    SUBDATASET_4_DESC=[24x170x180] sea_surface_temperature (32-bit
floating-point)Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0,  512.0)
Upper Right (  512.0,    0.0)
Lower Right (  512.0,  512.0)
Center      (  256.0,  256.0)
```

Ces fichiers NetCDF contiennent 4 jeux de données, *lon_bnds*, *lat_bnds*, *tim_bnds* et *tos*. Maintenant sélectionnez le sous-jeu de données, décrit comme :

```
NETCDF:"sst.nc":tos

[24x17x180] sea_surface_temperature (32-bit floating-point)
```

et obtenez l'information sur le nombre de bande qu'il y a à dans cette variable.

```
$ gdalinfo NETCDF:"sst.nc":tos
Driver: netCDF/Network Common Data Format
Size is 180, 170
Coordinate System is `'
```

```
Origin = (1.000000,-79.500000)
Pixel Size = (1.98888889,0.99411765)
Metadata:
    NC_GLOBAL#title=IPSL  model output prepared for IPCC Fourth
Assessment SRES A2 experiment
    NC_GLOBAL#institution=IPSL (Institut Pierre Simon Laplace, Paris,
France)

.... D'autres métadonnées

    time#standard_name=time
    time#long_name=time
    time#units=days since 2001-1-1
    time#axis=T
    time#calendar=360_day
    time#bounds=time_bnds
    time#original_units=seconds since 2001-1-1
Corner Coordinates:
Upper Left  (   1.0000000, -79.5000000)
Lower Left  (   1.0000000,  89.5000000)
Upper Right (     359.000,     -79.500)
Lower Right (     359.000,      89.500)
Center      ( 180.0000000,   5.0000000)
    Band 1 Block=180x1 Type=Float32, ColorInterp=Undefined
    NoData Value=1e+20
    Metadata:
        NETCDF_VARNAME=tos
        NETCDF_DIMENSION_time=15
        NETCDF_time_units=days since 2001-1-1
Band 2 Block=180x1 Type=Float32, ColorInterp=Undefined
    NoData Value=1e+20
    Metadata:
        NETCDF_VARNAME=tos
        NETCDF_DIMENSION_time=45
        NETCDF_time_units=days since 2001-1-1

.... D'autres bandes

Band 22 Block=180x1 Type=Float32, ColorInterp=Undefined
    NoData Value=1e+20
    Metadata:
        NETCDF_VARNAME=tos
        NETCDF_DIMENSION_time=645
        NETCDF_time_units=days since 2001-1-1
Band 23 Block=180x1 Type=Float32, ColorInterp=Undefined
    NoData Value=1e+20
    Metadata:
        NETCDF_VARNAME=tos
        NETCDF_DIMENSION_time=675
        NETCDF_time_units=days since 2001-1-1
Band 24 Block=180x1 Type=Float32, ColorInterp=Undefined
    NoData Value=1e+20
    Metadata:
        NETCDF_VARNAME=tos
        NETCDF_DIMENSION_time=705
        NETCDF_time_units=days since 2001-1-1
```

`gdalinfo` affiche le nombre de bandes dans un sous-jeu de données. Il y a des méta-données attachées à chaque bande. Dans cet exemple, les méta-données indique que chaque bande corresponde à un tableau de température mensuelle de la surface de la mer à partir de janvier 2001. Il y a 24 mois de données dans ce sous-jeu de données. Vosu pouvez utiliser `gdal_translate` pour lire le sous-jeu de données.

Notez que vous devez fournir exactement le contenu de la ligne noté *SUBDATASET_n_NAME* à GDAL, incluant le préfixe *NETCDF:*.

Le préfixe *NETCDF:* doit être en premier. Il déclenche le pilote netCDF du sous-jeu de données. Ce pilote a pour objectif seulement pour importer de capteurs distant et des jeux de données géospatiales sous la forme d'image raster. Si vous voulez explorer toutes les données contenues dans le fichier NetCDF vous devez utiliser un autre outil.

B Dimension

Le pilote NetCDF suppose que les données suivent la convention CF-1 d'UNIDATA. Les dimensions dans les fichiers NetCDF utilisent les règles suivantes : (Z,Y,X). S'il y a plus de 3 dimensions, le pilote les fusionnera en bandes. Par exemple si vous avez un tableau à 4 dimensions de type (P, T, Y, X). Le pilote multipliera les 2 dernières dimensions (P*T). Le pilote affichera les bandes dans l'ordre suivant. Il incrémentera d'abord T puis P. Les méta-données seront affichées sur chaque bande avec ses valeurs T et P correspondantes.

C Géoréférencement

Il n'y a pas de manière universelle de stocker le géoréférencement dans les fichiers netCDF. Le pilote tente d'abord de suivre la convention CF-1 à partir d'UNIDATA en cherchant la méta-données nommé "*grid_mapping*". Si "*grid_mapping*" n'est pas présent, le pilote tentera de trouver un tableau de grille lat/lon pour définir le tableau de géoréférencement. Le pilote NetCDF vérifie que le tableau lat/lon est espacé équitablement.

Si ces deux méthodes échouent, le pilote NetCDF tentera de lire les méta-données suivantes directement et définira un géoéréférencement.

- spatial_ref (Well Known Text)
- GeoTransform (GeoTransform array)

ou,

- *Northernmost_Northing*
- *Southernmost_Northing*
- *Easternmost_Easting*
- *Westernmost_Easting*

D Problèmes de créations

Ce pilote gère la création de fichier netCDF en suivant la convention CF-1. Vous pouvez créer des ensembles de jeux de données 2D. Chaque tableau de variable est nommé Band1, Band2, ... BandN.

Chaque bande possédera des métadonnées liée en donnant une courte description de la donnée qu'elel contient.

E Méta-données GDAL pour NetCDF

Tous les attributs de netCDF sont traduits de manière transparente vers les méta-données GDAL.

La traduction suit les règles suivantes :

- Les méta-données de NetCDF global ont une balise préfixé *NC_GLOBAL*.
- Les méta-données du jeu de données ont leur noms de variable préfixés.
- Chaque préfixe est suivie du signe #.
- L'attribut NetCDF suit la forme : *name=value*.

Exemple :

```
$ gdalinfo NETCDF:"sst.nc":tos
Driver: netCDF/Network Common Data Format
Size is 180, 170
Coordinate System is `'
Origin = (1.000000,-79.500000)
Pixel Size = (1.98888889,0.99411765)
Metadata:
```

Les attributs globaux de NetCDF :

```
NC_GLOBAL#title=IPSL  model output prepared for IPCC Fourth Assessment
SRES A2 experiment
```

Les attributs des variables pour : tos, lon, lat et time

```
tos#standard_name=sea_surface_temperature
tos#long_name=Sea Surface Temperature
tos#units=K
tos#cell_methods=time: mean (interval: 30 minutes)
tos#_FillValue=1.000000e+20
tos#missing_value=1.000000e+20
tos#original_name=sosstsst
tos#original_units=degC
tos#history= At   16:37:23 on 01/11/2005: CMOR altered the data in the
following ways: added 2.73150E+02 to yield output units;  Cyclical
dimension was output starting at a different lon;
lon#standard_name=longitude
lon#long_name=longitude
lon#units=degrees_east
lon#axis=X
lon#bounds=lon_bnds
lon#original_units=degrees_east
lat#standard_name=latitude
lat#long_name=latitude
lat#units=degrees_north
lat#axis=Y
lat#bounds=lat_bnds
lat#original_units=degrees_north
time#standard_name=time
time#long_name=time
time#units=days since 2001-1-1
time#axis=T
time#calendar=360_day
```

```
time#bounds=time_bnds
time#original_units=seconds since 2001-1-1
```

F Compilation du pilote

Ce pilote est compilé avec la bibliothèque netCDF d'UNIDATA.

Vous devez télécharger ou compiler la bibliothèque netCDF avant de configurer GDAL avec la gestion de netCDF.

S'il vous plait, notez qu'avec CygWIN vous devez vous assurer que les DLL sont éxécutable ou bien GDAL ne se lancera pas.

```
chmod a+rx [NetCDF DLLs]
```

Le répertoire des DLL de netCDF doit être dans votre *PATH*.

G Voir également

- convention NetCDF CF-1.0
- Bibliothèque NetCDF compilé
- Documentation NetCDF

Chapitre LV NITF -- National Imagery Transmission Format

GDAL gère la lecture de plusieurs sous-types de fichiers images NITF, et l'écriture des fichiers NITF 2.1 simples. Les fichiers NITF 1.1, NITF 2.0, NITF 2.1 et NSIF 1.0 avec des images non compressées, compressé en ARIDPCM, JPEG, compression JPEG200 (avec le SDK Kakadu ou ECW) ou VQ devraient être lisible.

Le test de la gestion de la lecture a été réalisé sur différents produits, dont CIB et les frames CADRG des produits RPF, frames ECRG, produits HRE.

Les tables de couleurs pour les images en pseudo-couleur sont lu. Dans certains cas les valeurs *nodata* sont identifiées.

Les étendues Lat/Long sont lu à partir des informations IGEOLO dans l'en-tête de l'image si elle est disponible. Si une information de géoréférencement en lat/long en haute définition est disponible dans les données auxiliaires RPF, ceux-ci seront utilisé en préférence à l'information IGEOLO de faible précision. Au cas où une instance de BLOCKA est trouvée, les coordonnées de plus haute précision du BLOCKA sont utilisées si les données du bloc couvre l'image complète – c'est à dire si le champ L_LINES avec le nombre de ligne pour ce bloc est égal au nombre de ligne de l'image. De plus, toutes les instances du BLOCKA sont renvoyées comme méta-données. Si GeoSDE TRE est disponible, elle sera utilisé pour fournir des coordonnées de plus hautes précisions.

La plupart des fichiers d'en-tête et des champs d'en-tête de l'image sont renvoyé comme des méta-données de niveau du jeu de données.

A Problèmes lors de la création

À l'export, les fichiers NITF sont toujours écrit en NITF 2.1 avec une image et aucune autre couche auxiliaire. Les images sont non compressées par défaut, mais les compressions JPEG et JPEG2000 sont également disponible. Le géoréférencement peut seulement être écrit pour les images utilisant le système de coordonnées géographique ou une projection WGS84 UTM. Les coordonnées sont implicitement traité comme WGS84 même s'ils sont en réalité dans un système de coordonnée géographique différent. Les tables de pseudo-couleur peuvent être écrites pour les images 8 bites.

En plus de l'export orienté par l'API *CreateCopy()*, il est également possible de créer un

fichier NTIF vide en utilisant Create() et d'écrire l'image à la demande. Cependant, en utilisant cette méthodologie, l'écriture de tables pseudo-couleurs et le géoréférencement ne sont pas gérés à moins que des options IREP et ICORDS appropriées ne soient fournit.

Options de création :

La plupart des en-tête de fichier, des méta-données d'en-tête des images et des champs de sécurité peuvent être définie avec des **options de création** appropriées (bien qu'elles sont reportées comme item de métadonnées, mais ne doit pas être définir comme métadonnées). Par exemple définir "FTITLE=Image of abandoned missle silo south west of Karsk" dans la liste d'option de création entraînera la configuration du champ FTITLE dans l'en-tête du fichier NITF. Utilisez les noms des champs officiels à partir du document de spécification NITF ; de placer pas le préfixe "NITF" qui est reporté lors de la demande de la liste de métadonnées.

- **IC=NC/C3/M3/C8 :** définie la méthode compression :
  - NC est la valeur par défaut, et signifie pas de compressions ;
  - C3 signifie compression JPEG et est seulement disponible pour la méthode `CreateCopy()`. Les options de création spécifique au format JPEG *QUALITY* et *PROGRESSIVE* peuvent être utilisé. Voyez la documentation sur le :ref:`gdal.gdal.formats.jpeg`. À partir de GDAL 1.7.0, les images multi-blocs peuvent être écrite.
    Unknown interpreted text role "ref".
  - M3 est une variation de C3. la seule différence est qu'une carte de bloc est écrite, ce qui permet la recherche rapide d'un bloc (à partir de GDAL 1.7.0).
  - C8 signifie compression JPEG2000 (un bloc) et est disponible pour les deux méthodes `CreateCopy()` et `Create()`. La compression *JPEG2000* est seulement disponible si le pilote *JP2ECW* est disponible. Les options de création spécifique au format JP2ECW *TARGET* et *PROFILE* peuvent être utilisée. Voyez la documentation du :ref:`gdal.gdal.formats.jp2ecw`. À partir de GDAL 1.7.0, si le pilote JP2ECW n'est pas disponible, le pilote Jasper JPEG2000 peut être utilisé dans le cas de `CreateCopy()`.
    Unknown interpreted text role "ref".
- **NUMI=n :** (à partir de GDAL 1.7.0) Nombre d'images. 1 par défaut. Cette option est seulement compatible avec IC=NC (images non compressées).
- **ICORDS=G/D/N/S :** définir à "G" pour s'assurer que l'espace sera réservé pour les coordonnées du coin géographique (en DMS) pour être définie plus tard par `SetGeoTransform()`, définie à "D" pour les coordonnées géographiques en degré décimal, définie à "N" pour la projection UTM WGS84 dans l'hémisphère nord ou à "S" pour la projection UTM WGS84 dans l'hémisphère sud (seulement nécessaire pour la méthode `Create()`, pas pour `CreateCopy()`). Si vous créez un nouveau fichier NITF et avez définie "N" ou "S" pour ICORDS, vous devez appeler plus tard la méthode *SetProjection* avec un SRS UTM cohérent pour définir un numéro de zone UTM (autrement elle sera à 0 par défaut).
- **FHDR :** la version du fichier peut être sélectionnée bien que les deux seules possibilités gérées sont "NITF02.10" (celui par défaut), et "NSIF01.00".
- **IREP :** définir à "RGB/LUT" pour réserver l'espace pour une table de couleur pour chaque bande en sortie (seulement nécessaire pour la méthode `Create()`, pas pour `CreateCopy()`).
- **LUT_SIZE :** définie pour contrôler la taille des tables de pseudo-couleurs pour les bandes RVB/LUT. Une valeur de 256 est assumé si aucune est présente (seulement nécessaire pour la méthode `Create()`, pas pour `CreateCopy()`).
- **TFW=YES :** force la génération d'un fichier world ESRI associé (.tfw).
- **BLOCKXSIZE=n :** définie la largeur du bloc.

- **BLOCKYSIZE=n :** définie la hauteur du bloc.
- **BLOCKA_*= :** si un ensemble d'option de BLOCKA est fournit avec la même organisation que la méta-donnée NITF_BLOCKA rapporté lors de la lecture d'un fichier NITF avec des arbres BLOCKA, alors un fichier sera créer avec des arbres BLOCKA.
- **TRE=tre-name,tre-contents :** un ou plusieurs options de création d'arbres peuvent être utilisées pour écrire des arbres définie par l'utilisateur à l'en-tête de l'image. Le nom de l'arbre doit être de moins de 6 caractères, et le contenu de l'arbre doit être protégé par un s'il contient des ou des bytes à 0. L'argument est le même format que celui renvoyé dans le domaine méta-données de l'arbre lors de la lecture.
- **FILE_TRE=tre-name=tre-contents :** (GDAL >= 1.8.0) similaire aux options ci-dessus, sauf que TRE sont écrite dans l'en-tête du fichier, au lieu de l'en-tête de l'image.
- **SDE_TRE=YES/NO :** (GDAL >= 1.8.0) écrite GEOLOB et GEOPSB TREs pour obtenir un géoréférencement plus précis. Ceci est limité au SRS géographique, et pour `CreateCopy()` pour le moment.

B Liens

- :ref:`gdal.gdal.formats.nitf_avancee`
  Unknown interpreted text role "ref".
- Page publique du Bureau Technique du NITFS.
- DIGEST Part 2 Annex D (describe encoding of NITF Spatial Data Extensions).
- :ref:`gdal.gdal.formats.divers_formats.rpftoc` : pour lire la Table Of Content des produits CIB et CADRG.
  Unknown interpreted text role "ref".
- MIL-PRF-89038 : Spécification des produits RPF, CADRG, CIB.
- :ref:`gdal.gdal.formats.divers_formats.ecrgtoc` : pour lire la Table Of Contents des produits ECRG.
  Unknown interpreted text role "ref".
- Duplicate explicit target name: "mil-prf-32283".
  MIL-PRF-32283 : Spécification des produits ECRG.

C Crédit

L'auteur souhaite remercier AUG Signal (http://www.augsignals.com/) et le programme GeoConnections (http://geoconnections.org/) pour l'aide au développement de ce pilote ainsi que Steve Rawlinson (JPEG), Reiner Beck (BLOCKA) pour l'aide à l'ajout de fonctionnalités.

Chapitre LVI NITF -- Information Avancée sur le pilote

Le pilote NITF (*National Imagery Transmission Format*) dans GDAL inclue un nombre d'options avancée et plus ou moins ésotérique ne convient pas à la documentation générale de l'utilisation pour le pilote. Cette information est collecté ici et est surtout utilisable pour les développeurs et les utilisateurs avancés.

A Segments CGM

Les fichiers NITF qui ont des données CGM (qui sont un segment de type GR - graphique, ou SY avec un STYPE de valeur 'C') rendront cette information disponible comme méta-données dans le domaine CGM. Les méta-données renvoyées ressemblera à cela :

```
SEGMENT_COUNT=1
SEGMENT_0_SLOC_ROW=25
SEGMENT_0_SLOC_COL=25
SEGMENT_0_SDLVL=2
SEGMENT_0_SALVL=1
SEGMENT_0_SLOC_ROW=25
SEGMENT_0_SLOC_COL=25
SEGMENT_0_SDLVL=2
SEGMENT_0_SALVL=1
SEGMENT_0_CCS_ROW=00025
SEGMENT_0_CCS_COL=00025
SEGMENT_0_DATA= \0!\0...
```

Les valeurs SLOC_ROW et SLOC_COL sont l'emplacement de l'objet CGM relatif à l'image de base (SALVL). Les valeurs CCS_ROW/COL sont relatives au système de coordonnées commun. _SDLVL est le niveau d'affichage. DATA est les données brutes CGM avec une protection par des appliquées. Toutes les occurrence de zéro ASCII sera traduit en "0", et tous les symboles et " seront protégés avec . La fonction `CPLUnescapeString()` peut être utilisé pour protégé les données avec / en format binaire en utilisant le schéma *CPLES_BackslashQuotable*.

À partir de 1.8.0, pour ajouter des données CGM à une image NITF, vous pouvez passer des options de création dans le format suivant :

```
CGM=SEGMENT_COUNT=1
CGM=SEGMENT_0_SLOC_ROW=25
CGM=SEGMENT_0_SLOC_COL=25
CGM=SEGMENT_0_SDLVL=2
CGM=SEGMENT_0_SALVL=1
CGM=SEGMENT_0_DATA=\0!\0...
```

Notez que passer CGM comme options de création écrasera le segment CGM lu dans le domaine de métadonnées CGM.

Bien que GDAL ne gère pas le parsage ou le rendu de données CGM, au moins un utilisateur a trouvé la bibliothèque UniConverter utile pour cela.

B Fichiers NITF Multi-Image

Les fichiers NITF avec plus d'un segment d'image (IM) présentera les segments d'image comme des sous jeux de données. L'ouverture de multiple fichiers NITF par noms de fichier fournira un accès au premier segment d'image. Les méta-données des sous jeux de données pour les trois fichiers NITF images ressemblera à cela :

```
Subdatasets:
    SUBDATASET_1_NAME=NITF_IM:0:multi_image_jpeg_2.0.ntf
    SUBDATASET_1_DESC=Image 1 of multi_image_jpeg_2.0.ntf
    SUBDATASET_2_NAME=NITF_IM:1:multi_image_jpeg_2.0.ntf
    SUBDATASET_2_DESC=Image 2 of multi_image_jpeg_2.0.ntf
    SUBDATASET_3_NAME=NITF_IM:2:multi_image_jpeg_2.0.ntf
    SUBDATASET_3_DESC=Image 3 of multi_image_jpeg_2.0.ntf
```

Dans ce cas l'ouverture de *multi_image_jpeg_2.0.ntf* directement donnera un accès à *NITF_IM:0:multi_image_jpeg_2.0.ntf*. Pour ouvrir les autres utilisez les noms des sous jeux de données correspondant. Le mécanisme de sous jeu de données est un concept GDAL générique discuté dans le document Modèle de données.

C Segments Texte

Les fichiers NITF qui ont des segments textes (qui est un segment de type TX) rendra cette information disponible comme méta-données dans le domaine TEXT. La méta-données renvoyée ressemblera à :

```
EADER_0=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
DATA_0=This is test text file 01.

HEADER_1=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
DATA_1=This is test text file 02.

HEADER_2=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
DATA_2=This is test text file 03.

HEADER_3=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
DATA_3=This is test text file 04.

HEADER_4=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
DATA_4=This is test text file 05.
```

L'argument à DATA_n est le texte brute du n ième (à partir de 0) segment texte avec aucune protection de quelque forme que ce soit appliquée.

À partir de GDAL 1.8.0, les données d'en-tête du segment TEXT sont préservé dans l'item des métadonnées HEADER_n.

La méthode `CreateCopy()` sur le pilote NITF gère également la création de segments texte sur le fichier de sortie aussi longtemps que le fichier en entré possède des méta-données dans le domaine TEXT comme définie ci-dessus.

À partir de GDAL 1.8.0, pour ajouter des données TEXT à une image NITF, vous pouvez aussi passer les options de création dans le format suivant :

```
TEXT=HEADER_0=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
TEXT=DATA_0=This is test text file 01.
TEXT=HEADER_1=TE       00020021216151629xxxxxxxxxxxxxxxxxxxxxxxxxx
TEXT=DATA_1=This is test text file 02.
```

Notez que passer TEXT comme option de création écrasera le texte existant lu dans le domaine de métadonnées TEXT.

D TRE

Les fichiers NITF avec des extensions enregistrées (ou non enregistrées ?) sur l'en-tête du fichier, ou l'en-tête de l'image géoréférencée les rendront disponible sous une forme brute dans les méta-données via le domaine TRE. Le domaine TRE contiendra une méta-données par TRE qui aura le nom du TRE comme nom, et la données du TRE comme contenu. La donnée contenue sera protégé par comme les données CGM ci-dessus.

Dans le cas d'occurrences multiples du même TRE, la seconde occurrence sera nommée "TRENAME_2", le troisième "TRENAME_3" où TRENAME est le *nom TRE*.

```
Metadata (TRE):
    GEOPSB=MAPM  World Geodetic System 1984
           WGE World Geodetic System 1984
               WE Geodetic
                  GEODMean Sea
                      MSL 000000000000000
                                          0000
PRJPSB=Polar Stereographic
     PG2-00090.00000250000039.99999884000000000000000000000000000000
MAPLOB=M  0598005958-000003067885.8-000002163353.8
```

E TREs comme xml:TRE

À partir de GDAL 1.9.0, tous les TRE trouvé dans le fichier et correspondant à l'une des descriptions de TRE du fichier nitf_spec.xml dans le répertoire données de GDAL seront reporté comme contenu XML dans le domaine de métadonnées xml:TRE.

```
Metadata (xml:TRE):
<tres>
<tre name="RSMDCA" location="des TRE_OVERFLOW">
    <field name="IID" value="2_8" />
```

```
    <field name="EDITION" value="1101222272-2" />
    <field name="TID" value="1101222272-1" />
    <field name="NPAR" value="06" />
    <field name="NIMGE" value="001" />
    <field name="NPART" value="00006" />
    <repeated name="IMAGE" number="1">
    <group index="0">
        <field name="IID" value="2_8" />
        <field name="NPARI" value="06" />
    </group>
    </repeated>
    <field name="XUOL" value="-2.42965895449297E+06" />
    <field name="YUOL" value="-4.76049894293300E+06" />
    <field name="ZUOL" value="+3.46898407315533E+06" />
    <field name="XUXL" value="+8.90698769551156E-01" />
    <field name="XUYL" value="+2.48664813021570E-01" />
    <field name="XUZL" value="-3.80554217799520E-01" />
    <field name="YUXL" value="-4.54593996792805E-01" />
    <field name="YUYL" value="+4.87215943350720E-01" />
    <field name="YUZL" value="-7.45630553709282E-01" />
    <field name="ZUXL" value="+0.00000000000000E+00" />
    <field name="ZUYL" value="+8.37129879594448E-01" />
    <field name="ZUZL" value="+5.47004172461403E-01" />
[...]
    <repeated name="DERCOV" number="21">
    <group index="0">
        <field name="DERCOV" value="+5.77388827727787E+04" />
    </group>
[...]
    <group index="20">
        <field name="DERCOV" value="+1.14369570920252E-02" />
    </group>
    </repeated>
</tre>
<tre name="RSMECA" location="des TRE_OVERFLOW">
[...]
</tre>
<tre name="RSMIDA" location="des TRE_OVERFLOW">
[...]
</tre>
<tre name="RSMPCA" location="des TRE_OVERFLOW">
[...]
</tre>
</tres>
```

F Fichier brute/ En-tête d'image

Dans certains cas l'application peut avoir besoin de récupérer des informations très spécifique à partir de l'image ou de l'en-tête du fichier qui n'est pas disponible normalement comme métadonnées. Dans ce cas il est possible d'interroger le domaine de métadonnées "NITF_METADATA". Le fichier complet et les en-têtes d'image seront renvoyés comme métadonnées au format encodé en base64. Quelque chose comme :

```
Metadata (NITF_METADATA):
```

```
NITFFileHeader=002213 TklURjAyLjAwMDEgICAgVTIxN0cwSjA...
NITFImageSubheader=439 SU1NaXNzaW5nIElEMjUxNTI1NTlaTU...
```

Notez que les valeurs numériques encodées en ascii préfixant l'en-tête encodé en base64 est la longueur (décodé) en octets, suivit d'une espace.

Chapitre LVII OGDI -- Pont OGDI

Warning!

À partir de GDAL >= 1.5.0, il y a très peu de raison d'utiliser le pont vers le pilote OGDI raster puisque les formats :ref:`gdal.gdal.formats.divers_formats.adrgarc`, :ref:`gdal.gdal.formats.dted` et :ref:`gdal.gdal.formats.divers_formats.rpftoc` (CADRG/CIB) sont gérés nativement par GDAL.

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

Les sources de données raster OGDI sont géré par GDAL en lecture. À la fois les matrices et les familles devraient être géré, ainsi que la lecture des cartes de couleurs et les méta-données des projections. Le lecteur GDAL a pour but d'être utilisé avec le pilote OGDI 3.1, mais les pilotes OGDI 3.0 devrait aussi fonctionner.

Les jeux de données OGDI sont ouvert dans GDAL par la sélection de l'url GLTP. Par exemple, *gltp://gdal.velocet.ca/adrg/usr4/mpp1/adrg/TPSUS0101* ouvrira le jeu de données ADRG stocké dans /usr4/mpp1/adrg/TPSUS0101 sur la machine gdal.velocet.ca (en assumant qu'il y ait un serveur OGDI en fonctionnement) en utilisant le pilote 'adrg'. Cet accès par défaut à l'ensemble du serveur de données tentera de représenter toutes les couches (et tous les types de famille) comme des bandes, toutes à la résolution et à la région rapporté par le serveur de données lors de l'accès initial.

Il est également possible desélectionner une couche particulière et d'accéder à la famille d'un serveur de données OGDI en indiquant le nom de la couche dans le nom. Le nom du jeu de données GDAL `gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0102.IMG"` : la matrice sélectionnera la couche nommée TPUS0102.IMG du jeu de données /usr4/mpp1/adrg/TPUS0101 sur le système local en utilisant le pilote ADRG, et accèdera à la famille de la matrice. Quand on accède à une couche spécifique decette manière, GDAL tentera de déterminer la région et la résolution à partir du document capabilities OGDI 3.1. Notez que les serveurs de données OGDI 3.0 doivent avoir la couche et la famille définie dans le nom du jeu de données puisqu'ils ne peuvent pas être déterminer automatiquement.

Par exemple :

```
gltp://gdal.velocet.ca/adrg/usr4/mpp1/adrg/TPUS0101
gltp:/adrg/usr4/mpp1/adrg/TPUS0101
gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0102.IMG":Matrix
```

Les couches de famille Matrix OGDI (couches d'entier pseudo-couleur) sont représentées comme une simple bande de raster avec une table de couleur. Bien que les couches Matrix contiennent des valeurs entières de 32 bits, elles sont représentées dans GDAL comme 8 couches. Toutes les valeurs au-dessus de 255 sont tronqué à 255, et seulement les 256 entrées de la table de couleur sont capturées. Puisque cela fonctionne bien pour les couches Matrix, il est espéré que les couches Matrix avec un domaine plus dynamique soient représenté dans un futur proche sous forme d'un autre type de données.

Les couches de famille d'Image OGDI peuvent en interne avoir un type RVB (1) qui est représenté sous forme de trois bandes dans GDAL, ou en Byte (2), UInt16 (3), Int16 (4) ou Int32 (5). Il n'y a pas de gestion pour les bandes de virgules flottantes dans OGDI 3.1.

Le pilote OGDI de GDAL représentera les sources de données OGDI comme ayant des aperçues *arbitraires*. N'importe quel raster de GDAL lu par des requêtes à une résolution réduite sera passé au pilote OGDI avec cette résolution réduite ; permettant potentiellement une efficacité dans la lecture des informations des aperçues à partir d'un serveur de données OGDI.

Si un serveur de données OGDI est ouvert sans avoir sélectionner le nom d'une couche dans le nom du jeu de données, et si le serveur de données a les capacités des styles OGDI 3.1, la liste des couches sera rendu disponible comme des méta-données de sous-jeu de données. Par exemple, la commande `gdalinfo` pourra renvoyer ce qui suit. Ces informations peuvent être utilisé pour établir les couches disponibles pour un accès direct.

```
Subdatasets:
    SUBDATASET_1_NAME=gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0101.IMG
":Matrix
    SUBDATASET_1_DESC=TPUS0101.IMG as Matrix
    SUBDATASET_2_NAME=gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0102.IMG
":Matrix
    SUBDATASET_2_DESC=TPUS0102.IMG as Matrix
    SUBDATASET_3_NAME=gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0101.IMG
":Image
    SUBDATASET_3_DESC=TPUS0101.IMG as Image
    SUBDATASET_4_NAME=gltp:/adrg/usr4/mpp1/adrg/TPUS0101:"TPUS0102.IMG
":Image
    SUBDATASET_4_DESC=TPUS0102.IMG as Image
```

Lisez également :

- ogdi.sourceforge.net

Chapitre LVIII OZI -- raster OZF2/OZFX3

(Disponible à partir de GDAL >= 1.8.0)

GDAL gère la lecture des jeux de données raster OZF2/OZFX3.

Soit le fichier image soit le fichier .map peut être passé à GDAL. Pour récupérer le géoréférencement, vous devez spécifier le fichier .map.

A Voir également

- Spécification du format OZF2/OZFX3

Chapitre LIX PCIDSK --- PCI Geomatics Database File

Fichiers de base de données PCIDSK utilisé par les logiciels PCI EASI/PACE pour l'analyse d'image. Il est géré en lecture et écriture par GDAL. Tous les types de données de pixel, et d'organisation des donnés (pixel entrelacé, bande entrelacée, fichier entrelacé et tuile) doivent être gérés.

Pour l'instant, les segments LUT et PCT sont ignorés, mais les segments PCT doivent être traité comme associé avec les bandes. Les fichiers d'ensemble et les méta-données spécifique aux bandes doivent être correctement associés avec l'image ou les bandes.

Le géoréférencement est géré bien qu'il peut y avoir certaines limitations dans la gestion des *datums* et des ellipsoïdes. Les segments des points d'amer sont ignorés. les segments RPC seront renvoyés comme métadonnées RPC dans le style de GDAL

Les aperçues internes d'images (pyramide) seront également lu correctement bien que les nouveaux aperçues demandés seront construit en externe comme un fichier .ovr.

Les segments vectoriels sont géré par le pilote OGR PCIDSK.

A Options de création

Notez que les fichiers PCIDSK ont toujours produit des pixels entrelacés, même si d'autres organisations sont géré en lecture.

- **INTERLEAVING=PIXEL/BAND/FILE/TILED :** définie l'entrelacement pour les données des fichiers raster.
- **COMPRESSION=NONE/RLE/JPEG :** définie la compression à utiliser. Les valeurs autre que NONE (celle par défaut) peut être seulement utilisé avec l'entrelacement TILED. Si JPEG est sélectionné il peut inclure une valeur de qualité comprise entre 1 et 100 - par exemple COMPRESSION=JPEG40.
- **TILESIZE=n :** quand INTERLEAVING est TILED, la taille des tuiles peut être sélectionné avec ce paramètre - par défaut 127 pour 127x127.

B Lisez également

- Implémenté dans *gdal/frmts/pcidsk/pcidskdataset2.cpp*.
- SDK PCIDSK.

Chapitre LX Geospatial PDF

(Disponible à partir de GDAL >= 1.8.0)

GDAL gère la lecture des documents PDF géospatial, en extrayant les informations géoréférencement et rasterise les données. Les documents PDF non géospatial seront aussi reconnu par le pilote.

GDAL doit compiler avec la gestion de libpoppler (licencé en GPL), et libpoppler lui même doit avoir été configuré avec `--enable-xpdf-headers` afin que les en têtes xpdf C++ soient disponibles. Note : l'API C++ poppler n'est pas stable, la compilation du pilote peut donc échouer avec des versions trop ancienne ou trop récente. Les versions testés avec succès sont poppler >= 0.12.X and <= 0.16.0.

À partir de GDAL 1.9.0, comme une alternative, le pilote PDF peut être compilé avec libpodofo (sous licence LGPL) pour éviter la dépendance avec libpoppler. Cela est suffisant pour obtenir les informations de géoréférencement. Cependant, pour obtenir l'imagerie, l'utilitaire pdftoppm qui vient avec la distribution poppler doit être disponible dans le PATH du système. Un fichier temporaire sera généré dans un répertoire déterminé par les options de configuration suivantes : *CPL_TMPDIR*, *TMPDIR* ou *TEMP* (dans cet ordre). Si aucun n'est définie, le répertoire courant sera utilisé. Testé avec succès avec les versions 0.8.4 et 0.9.1 de libpodofo.

Le pilote gère la lecture de géoréférencement encodées dans l'un des deux moyens existants acutellemnt : en fonction des meilleures pratiques d'encodage de l'OGC, ou selon le supplément d'Adobe de la norme ISO 32000.

Les dimensions des raster peuvent être contrôlé en définissant le DPI de la rasterisation avec l'option de configuration *GDAL_PDF_DPI*. Sa valeur par défaut est 150.

Les documents de plusieurs pages sont exposés comme sous jeux de données, un sous jeu de données par page du document.

Les encarts (pour les meilleurs pratiques de l'OGC) ou les bounding box (style Adobe) seront reportés comme des items de métadonnées NEATLINE, afin qu'il puisse être utilisé plus tard comme une ligne de découpe pour l'algorithme de déformation.

À partir de GDAL 1.9.0, les métadonnées XMP peuvent être extraites du fichier, et seront stockées comme contenu brute XML dans le domaine de métadonnées xml:XMP.

A Restrictions

L'ouverture d'un document PDF (pour obtenir le géoréférencement) est rapide, mais au premier accès à un bloc raster, la page entière sera rasterisée, ce qui peut être une opération lente.

Seuls quelques-uns des systèmes de référence possibles disponibles dans les spécifications des meilleures pratiques de l'OGC ont été actuellement mappée dans le pilote. Les systèmes de référence non reconnus seront considérés comme étant basé sur l'ellipsoïde WGS84.

Pour les documents qui contiennent plusieurs lignes ordonnées dans une page (encart), le géoréférencement sera extrait de l'encart qui aura la plus grande superficie (en terme de points sur l'écran).

Il n'y a pour l'instant aucune gestion de sélection du rendu de couche.

B Voir également

- Bonne pratique de l'encodage GeoPDF de l'OGC version 2.2
- Supplément d'Adobe pour l'ISO 32000
- Homepage de Poppler
- Quelques échantillons PDF Geospatial
- D'autres échantillon PDF Geospatial

Chapitre LXI PDS -- Système de données planétaire

PDS est un format utilisé d'abord par la NASA pour stocker et distribuer des données images planétaire, lunaire et solaire. GDAL fournie un accès en lecture seul aux données d'imagerie formaté en PDS.

Les fichiers PDS ont souvent l'extension .img, parfois avec un fichier .lbl (label) associé. Quand un fichier .lbl file existe il doit être utilisé comme nom de jeu de données plutôt que celui du fichier .img.

En plus de la gestion de la plupart des configuration d'image PDS, ce pilote lit également les informations du système de coordonnées et de géoréférencement ainsi que d'autres métadonnées d'en-tête.

L'implémentation de ce pilote a été financé par la Surveillance Géologique des États-Unis.

En raison des ambiguïtés dans la spécification du PDS, le géoréférencement de certains produits est subtilement ou grossièrement incorrect. Il y a des variables de configuration qui peuvent être définies pour ces produits afin de corriger l'interprétation du géoréférencement. Certains détails sont disponibles dans le ticket #3940.

PDS fait partie de la famille des formats incluant ISIS2 et ISIS3.

A Voir également

- Implémentéd dans *gdal/frmts/pds/pdsdataset.cpp*.
- Système de données planétaire de la NASA
- :ref:`gdal.gdal.formats.isis2`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.isis3`
  Unknown interpreted text role "ref".

Chapitre LXII ISIS2 -- Cube ISIS de l'astrogéologie de l'USGS (Version 2)

ISIS2 est un format utilisé par le groupe Planetary Cartography de l'USGS pour stocker et distribuer des images planétaire. GDAL fournie un accès en lecture seule aux données images formaté en ISIS2.

Les fichiers ISIS2 ont souvent une extensions .cub, parfois avec un fichier .lbl (label) associé. Quand un fichier .lbl existe il doit être utilisé comme le nom du jeu de données plutôt que celui du fichier .cub.

En plus de la gestion pour la plupart des configurations d'images de ISIS2, ce pilote lit également les informations du système de coordonnées et de géoréférencement ainsi que d'autres métadonnées d'en-tête sélectionné.

L'implémentation de ce pilote a été financé par la *Geological Survey* des États-Unis.

ISIS2 fait partie de la famille des formats PDS et ISIS3.

A Problèmes de création

Pour le moment le pilote ISIS2 écrit un en-tête très minimal avec seulement les informations de structure de l'image. Aucun système de coordonnées, de géoréférencement ou d'autres métadonnées n'est capturé.

1 Options de création

- **LABELING_METHOD=ATTACHED/DETACHED :** Détermine si l'en-tête des étiquettes doivent être dans le même fichier que l'image (ATTACHED par défaut) ou dans un fichier séparé (DETACHED).
- **IMAGE_EXTENSION=extension :** définie l'extension utilisée pour les fichiers images détachés, "cub" par défaut. Utilisé seulement si LABELING_METHOD=DETACHED.

B Voir également

- Implémenté dans *gdal/frmts/pds/isis2dataset.cpp*.
- :ref:`gdal.gdal.formats.pds`

Unknown interpreted text role "ref".

- :ref:`gdal.gdal.formats.isis2`
  Unknown interpreted text role "ref".

Chapitre LXIII ISIS3 -- USGS Astrogeology ISIS Cube (Version 3)

ISIS3 est un format utilisé par le groupe Planetary Cartography de l'USGS pour stocker et distribuer des images planétaire. GDAL fournie un accès en lecture seule aux données images formaté en ISIS3.

Les fichiers ISIS3 ont souvent une extensions .cub, parfois avec un fichier .lbl (label) associé. Quand un fichier .lbl existe il doit être utilisé comme le nom du jeu de données plutôt que celui du fichier .cub.

En plus de la gestion pour la plupart des configurations d'images de ISIS3, ce pilote lit également les informations du système de coordonnées et de géoréférencement ainsi que d'autres métadonnées d'en-tête sélectionné.

L'implémentation de ce pilote a été financé par la *Geological Survey* des États-Unis.

ISIS3 fait partie de la famille des formats PDS et ISIS2.

A Voir également

- Implémenté dans *gdal/frmts/pds/isis2dataset.cpp*.
- :ref:`gdal.gdal.formats.pds`
  Unknown interpreted text role "ref".
- :ref:`gdal.gdal.formats.isis2`
  Unknown interpreted text role "ref".

Chapitre LXIV R -- R Object Data Store

Le format de fichier des objets R est pris en charge pour un accès en écriture, et un accès limité en lecture par GDAL. Ce format est le format natif que R utilise pour les objets sauvegardés avec la commande *save* et chargés commande *load*. GDAL gère l'écriture un jeu de données comme objet de tableau dans ce format, et supporte la lecture des fichiers avec les rasters simples essentiellement dans la même organisation. Il ne lira pas la plupart des fichiers objet R.

Pour le moment il n'y a pas de gestion de lecture et d'écriture d'informations géoréférencement.

A Options de création

- **ASCII=YES/NO :** produit un fichier formaté en ASCII, au lieu du binaire, si définie à YES. NO par défaut.
- **COMPRESS=YES/NO :** Produit un fichier compressé si YES, autrement un fichier non compressé. YES par défaut.

B Voir également

- Project R

Chapitre LXV Pilote GDAL Rasdaman

Rasdaman est un middleware de bases de données raster offrant un langage de requête de style SQL sur des tableaux multi-dimensionnels de taille illimitée, stockée dans une base de données relationnelle. Voir www.rasdaman.org pour le code open-source, la documentation, etc. Actuellement rasdaman est en considération pour incubation à l'OSGeo.

Dans notre implémentation du pilote, GDAL se connecte à rasdaman en définissant un modèle de requête qui est instancié avec la boîte concrète de sous-ensembles lors de chaque accès. Cela permet de livrer des découpes 2-D à partir de données n-D définie (tel que les série temporelle de données satellites hyperspectraux, des données de simulation climatique multi-variable, des données de modélisation océaniques, etc.). En particulier, l'imagerie virtuelle peut être proposés qui est dérivée à la demande des données de réelle du terrain. Des détails techniques supplémentaires sont donnés ci-dessous.

La syntaxe de la chaîne de connexion suit le motif WKT du raster et se pratique comme cela :

```
rasdaman:
    query='select a[$x_lo:$x_hi,$y_lo:$y_hi] from MyImages as a'
    [tileXSize=1024] [tileYSize=1024]
    [host='localhost'] [port=7001] [database='RASBASE']
    [user='rasguest'] [password='rasguest']
```

La chaîne de langage des requêtes rasdaman (rasql) dans ce cas s'effectue seulement en sous ensemble. Sur l'accès aux images par GDAL, le paramètre $ est substitués par la bounding box concrète calculé à partir des coordonnées des tuiles en entrée.

Toutefois, la requête fournis peut inclure tout type de traitement, tant qu'il renvoie quelque chose en 2-D. Par exemple, ceci détermine la moyenne des pixels rouges et le proche infrarouge de la série chronologique la plus ancienne image :

```
query='select ( a.red+a.nir ) /2 [$x_lo:$x_hi,$y_lo:$y_hi, 0 ] from
SatStack as a'
```

Pour le moment il n'y a pas de gestion de lecture et d'écriture d'informations géoréférencement.

A Voir également

- Project Rasdaman

Chapitre LXVI Rasterlite - Rasters in SQLite DB

À partir de GDAL 1.7.0, le pilote Rasterlite permet la lecture et la création de bases de données Rasterlite.

Ces bases de données peuvent être produites par les utilitaires de la distribution rasterlite, tel que rasterlite_load, rasterlite_pyramids, ...

Le pilote gère la lecture d'images en nuances de gris, en palette et RVB sous forme de tuiles GIF, PNG, TIFF ou JPEG.

Le pilote gère également la lecture des aperçues/pyramides, le système de référence spatiale et l'étendue spatiale.

Les tuiles compressé en onde ne sont pas gérées par défaut par GDAL, sauf si le pilote :ref:`gdal.gdal.formats.epsilon` a été compilé.

Unknown interpreted text role "ref".

GDAL/OGR doit être compilé avec la gestion du pilote SQLite. Pour la gestion de la lecture, la liaison avec la bibliothèque spatialite n'est pas nécessaire, mais une bibliothèque sqlite3 suffisamment récente est nécessaire pour lire les bases de données rasterlite. La bibliothèque rasterlite n'est pas requises non plus. Pour la gestion de l'écriture d'une nouvelle table, la liaison avec la bibliothèque *est* nécessaire

Bien que la documentation de Rasterlite ne mentionne que le GIF, PNG, TIFF, JPEG et WAVELET (pilote EPSILON) comme formats de compression pour les tuiles, le pilote gère la lecture et l'écriture des tuiles internes dans n'importe quel format pris en charge par GDAL. De plus, le pilote Rasterlite permet aussi la lecture et l'écriture autant de bandes et autant de types de bandes que géré par le pilote pour les tuiles internes.

A Syntaxe de la chaîne de connexion en mode lecture

Syntaxe : `'rasterlitedb_name' or 'RASTERLITE:rasterlitedb_name[,table=raster_table_prefix] [,minx=minx_val,miny=miny_val,maxx=maxx_val,maxy=maxy_val] [,level=level_number]`

où :

- *rasterlitedb_name* est le nom du fichier de la base de données rasterlite.
- *raster_table_prefix* est le préfixe de la table raster à ouvrir. Pour chaque raster, Il y a deux tables SQLite correspondantes, suffixé avec _rasters et _metadata
- *minx_val,miny_val,maxx_val,maxy_val* définie une étendue personnalisée (exprimée en unité du système de coordonnées) pour le raster qui peut être différent de l'étendue par défaut.
- *level_number* est le niveau de pyramide/Aperçue à ouvrir, 0 étant la base de la pyramide.

B Problèmes de création

Le pilote peut créer une nouvelle base de données si nécessaire, créer une nouvelle table raster si nécessaire et copier un jeu de données source dans la table raster définie.

Si les données existent déjà dans la table raster, les nouvelles données seront ajoutées. Vous pouvez utiliser les options de création *WIPE=YES* pour effacer les données existantes.

Le pilote ne gère pas la mise à jour d'un bloc dans une table existante. Il peut seulement ajouter de nouvelles données.

Syntaxe pour le nom du jeu de données en sortie :
`'RASTERLITE:rasterlitedb_name,table=raster_table_prefix' or 'rasterlitedb_name'`

Il est possible de définir seulement le nom de la base de données comme de la forme plus haut, mais seulement si la base de données n'existe pas déjà. Dans ce cas le nom de la table raster sera la base du nom de la base de données elle-même.

1 Options de création

- **WIPE (=NO by default):** définie à YES pour supprimer toutes les données pré-existantes dans la table définie
- **TILED (=YES by default) :** définie à NO si le jeu de données source doit être écrit comme une tuile unique dans la table raster
- **BLOCKXSIZE=n:** définie la largeur de la tuile. 256 par défaut.
- **BLOCKYSIZE=n:** définie la hauteur de la tuile. 256 par défaut.
- **DRIVER=[GTiff/GIF/PNG/JPEG/EPSILON/...] :** nom du pilote GDAL à utiliser pour stocker les tuiles. GTiff par défaut.
- **COMPRESS=[LZW/JPEG/DEFLATE/...] :** (pilote GTiff) nom de la méthode de compression
- **PHOTOMETRIC=[RGB/YCbCr/...] :** (pilote GTiff) interprétation photométrique
- **QUALITY :** (pilote GTiff compressé JPEG, JPEG et WEBP) qualité JPEG/WEBP 1-100. 75 par défaut.
- **TARGET :** (pilote EPSILON) réduction de la taille cible comme pourcentage de l'original (0-100). 96 par défaut.
- **FILTER :** (pilote EPSILON) identifiant du filtre. 'daub97lift' par défaut.

C Aperçues

Le pilote gère la construction (si le jeu de données est ouvert en mode update) et la lecture des aperçues internes.

Si aucun aperçue interne n'est détecté, le pilote tentera d'utiliser des aperçues externes (fichiers .ovr).

D Exemples

- Accéder à une BdD rasterlite avec une table raster unique :

```
$ gdalinfo rasterlitedb.sqlite -noct
```

En sortie :

```
Driver: Rasterlite/Rasterlite
Files: rasterlitedb.sqlite
Size is 7200, 7200
Coordinate System is:
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",0.01745329251994328,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]]
Origin = (-5.000000000000000,55.000000000000000)
Pixel Size = (0.002083333333333,-0.002083333333333)
Metadata:
TILE_FORMAT=GIF
Image Structure Metadata:
INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (  -5.0000000,  55.0000000) (  5d 0'0.00"W, 55d
0'0.00"N)
Lower Left  (  -5.0000000,  40.0000000) (  5d 0'0.00"W, 40d
0'0.00"N)
Upper Right (  10.0000000,  55.0000000) ( 10d 0'0.00"E, 55d
0'0.00"N)
Lower Right (  10.0000000,  40.0000000) ( 10d 0'0.00"E, 40d
0'0.00"N)
Center      (   2.5000000,  47.5000000) (  2d30'0.00"E,
47d30'0.00"N)
Band 1 Block=480x480 Type=Byte, ColorInterp=Palette
Color Table (RGB with 256 entries)
```

- Lister une BdD de table multi-raster :

```
$ gdalinfo multirasterdb.sqlite
```

En sortie :

```
Driver: Rasterlite/Rasterlite
Files:
Size is 512, 512
Coordinate System is `'
Subdatasets:
    SUBDATASET_1_NAME=RASTERLITE:multirasterdb.sqlite,table=rast
er1
    SUBDATASET_1_DESC=RASTERLITE:multirasterdb.sqlite,table=rast
er1
```

```
    SUBDATASET_2_NAME=RASTERLITE:multirasterdb.sqlite,table=rast
er2
    SUBDATASET_2_DESC=RASTERLITE:multirasterdb.sqlite,table=rast
er2
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0,  512.0)
Upper Right (  512.0,    0.0)
Lower Right (  512.0,  512.0)
Center      (  256.0,  256.0)
```

- Accéder à une table raster dans une BdD de table multi-raster :

```
$ gdalinfo RASTERLITE:multirasterdb.sqlite,table=raster1
```

- Créer une nouvelle BdD rasterlite avec des données encodées en tuiles JPEG :

```
$ gdal_translate -of Rasterlite source.tif
RASTERLITE:my_db.sqlite,table=source -co DRIVER=JPEG
```

- Créer des aperçues internes :

```
$ gdaladdo RASTERLITE:my_db.sqlite,table=source 2 4 8 16
```

- Nettoyer des aperçues internes :

```
$ gdaladdo -clean RASTERLITE:my_db.sqlite,table=source
```

- Créer des aperçues externe dans un fichier .ovr :

```
$ gdaladdo -ro RASTERLITE:my_db.sqlite,table=source 2 4 8 16
```

E Voir également

- Page principale sur Spatialite et Rasterlite
- Manuel sur Rasterlite
- Howto sur Rasterlite
- Base de données échantillon

Chapitre LXVII RIK -- Swedish Grid Maps

Géré par GDAL en accès en lecture. Ce format est utilisé dans les cartes créées par la *swedish organization Lantmäteriet*. Gestion des versions 1, 2 et 3 du format RIK, mais seulement 8 bits par pixel.

Ce pilote est basé sur le travail réalisé par le projet TrikPanel (http://sourceforge.net/projects/trikpanel/).

- **Note :** implémenté dans *gdal/frmts/rik/rikdataset.cpp*.

Chapitre LXVIII RMF --- Raster Matrix Format

RMF est un format simple raster tuilé utilisé dans les SIG "Integration" et "Panorama". Le format lui-même a très peu de possibilité.

Il y a deux types de RMF appelé MTW et RSW. MTW gère les données entières 16 bites et les points flottant 32/64 bites dans un seul canal et a pour objectif de stocké les données MNT. RSW est un raster avec un objectif plus général, il gère un canal simple avec une carte de couleur ou trois canaux d'image RVB. Seul les données 8 bits peuvent être rangé dans un RSW. Un géoréférencement simple peut être fournit pour les deux types d'image.

A Méta-données

- **ELEVATION_MINIMUM :** valeur d'élévation minimum (seulement MTW).
- **ELEVATION_MAXIMUM :** valeur d'élévation maximum (seulement MTW).
- **ELEVATION_UNITS :** nom de l'unité pour les valeurs raster (seulement MTW). Peut être "m" (mètres), "cm" (centimètres), "dm" (décimètres), "mm" (millimètres).
- **ELEVATION_TYPE :** peut être soit (élévation absolue) soit 1 (élévation totale). Seulement MTW.

B Options création

- **MTW=ON :** force la génération de matrice MTW (RSW sera crée par défaut).
- **BLOCKXSIZE=n :** définie la largeur de la tuile, par défaut, définie à 256.
- **BLOCKYSIZE=n :** définie la hauteur de la tuile, par défaut, définie à 256.

Lisez également :

- Implémenté dans *gdal/frmts/rmf/rmfdataset.cpp*.
- Page principale de "Panorama" GIS : http://www.gisinfo.ru/index_en.htm

Chapitre LXIX RS2 -- RadarSat 2 XML Product

Ce pilote lira certain produits polarimétrique XML de RadarSat 2. En particulier, les produits complexes, et les produits détecté de magnitude de 16 bits.

Les produits XML de RadarSat 2 sont distribués avec un fichier primaire en XML appelé *product.xml* et un ensemble de fichiers de données de support XML avec l'imagerie réelle stockée dans des fichiers TIFF. Le pilote RS2 sera utilisé si le fichier *product.xml* ou le répertoire le contenant est sélectionné, et il peut traiter toutes les imageries sous la forme d'un jeu de données consistant.

Les produits complexes utilisent des fichiers TIFF « 32 bites typé void » qui ne sont pas lisible d'une manière compréhensible normalement. Le pilote RS2 prend soin de convertir cela en un format Cint16 interne utile.

Le pilote RS2 lit également les points de géolocation à partir du fichier *product.xml* et les représentent comme des points d'amer sur le jeu de données.

Il est très probable que le format International de RadarSat sera distribué avec d'autres sortes de jeux de données dans de ce format : cependant, pour l'instant ce pilote gère spécifiquement les produits polarimétriques RadarSat 2. Tous les autres seront ignorés, ou résultat en plusieurs erreurs *runtime*. Il est espéré que ce pilote peut être généralisé avec d'autres échantillons du produit en fonction de leur disponibilité.

Lisez également :

- RadarSat document RN-RP-51-27.

Chapitre LXX ESRI ArcSDE Raster

ArcSDE d'ESRI fournit une couche d'abstraction pour de nombreuses bases de données qui permettent le stockage de données raster. ArcSDE gère l'imagerie à n-bandes avec plusieurs bit de profondeur, et l'implémentation en cours du pilote de GDAL doit gérer autant de bande que vous pouvez lui fournir. ArcSDE gère le stockage des données LZW, JP2K, et non compressées, et les présente d'une manière transparente à travers son SDK de son API en C.

A Fonctionnalités du pilote GDAL du Raster d'ArcSDE

Le pilote aujourd'hui gère les fonctionnalités suivantes :

- Gestion de la lecture seulement.
- Information de transformation spatiale pour les rasters qui l'ont définie.
- Information des références des coordonnées.
- Interprétation des couleurs (palette pour les jeux de données avec une carte des couleurs, en échelle de gris autrement).
- Statistique des bandes si ArcSDE les a mis en cache, autrement GDAL les calculera.
- Gestion des aperçues (pyramide) d'ArcSDE.
- Données sur 1 bit, 4 bit, 8 bit, 16 bit, et 32 bit.
- Gestion de IReadBlock qui correspond à la représentation par ArcSDE de la données dans la base de données.
- SDK de ArcSDE 9.1 et 9.2. Les versions plus anciennes peuvent aussi fonctionner, mais n'ont pas été testées.

Le pilote ne gère pas aujourd'hui les fonctionnalités suivantes :

- L'écriture de jeux de données GDAL dans la base de données.
- Lecture importante, rapide, et en un seul passage de la base de données.
- lecture à partir du "Catalogues Raster d'ArcSDE d'ESRI ".
- masque NODATA.

B Considérations des performances

Le pilote raster d'ArcSDE gère actuellement seulement les méthodes de lecture de block. Chaque appel à cette méthode résulte en un requête pour un block de données raster pour chaque bandes de données dans le raster, et les requêtes en un seul passage pour

toutes les bandes pour un block ou une zone donnée n'est pas réalisé actuellement. Par conséquence cette approche résulte en une sur-utilisation du réseau. On espère que le pilote sera amélioré pour gérer les lectures en un seul passage dans un futur proche.

Le pilote raster d'ArcSDE ne devrait consommer seulement une connexion ArcSDE tout au long de l'existence du jeu de données. Chaque connexion à la base de données a une surcharge d'approximativement de 2 secondes, avec une surcharge additionnelle utilisée pour calculer les informations du jeu de données. Ainsi, l'utilisation du pilote dans des situations où il y a plusieurs ouverture et fermeture de jeux de données n'est pas censé être très performants. Bien que le SDK en C d'ArcSDE gère les threading et les fermetures (locking), le pilote raster d'ArcSDE pour GDAL n'utilise pas ces fonctionnalités. Ainsi, ce pilote doit être considéré threadsafe, et partager des jeux de données entre threads engendrera des résultats indéfinies (et souvent désastreux).

C Spécification du jeu de données

Les jeux de données SDE sont définies avec les informations suivantes :

```
SDE:sdemachine.iastate.edu,5151,database,username,password,fully.speci
fied.tablename,RASTER
```

- **SDE:** – ceci est le préfixe qui indique à GDAL d'utiliser ou non le pilote SDE
- **sdemachine.iastate.edu** – le nom du DNS ou adresse IP du serveur auquel on se connecte.
- **5151** – le numéro du port (5151 ou port:5151) ou l'entrée du service (typiquement esri_sde).
- **database** – la base de données à se connecter. Cela peut aussi être vide et définie comme ...
- **username** – nom utilisateur.
- **password** – mot de passe.
- **fully.specified.tablename** – il est prudent d'utiliser un nom de table définie entièrement autant que possible, bien que cela ne soit absolument pas obligatoire.
- **RASTER** – nom optionnel de la colonne raster.

Chapitre LXXI Terragen --- Terragen™ Terrain File

Les fichiers Terragen terrain stockent des valeurs d'élévations sur 16 bites avec une grille d'espacement optionnelle (mais pas de positionnement). L'extension du fichier pour les heightfields de Terragen est "TER" ou "TERRAIN" (qui dans le cas précédent est le même que Leveller, mais le pilote ne reconnait que les fichiers Terragen). L'ID du pilote est "Terragen". Le jeu de données est basé sur des fichiers et possède seulement une bande d'élévation. Les élévations vides ne sont pas gérées. Les pixels sont considéré comme des points.

A Lecture

`dataset::GetProjectionRef()` renvoie un système de coordonnées local en utilisant des mètres.

`band::GetUnitType()` renvoie des mètres.

Les élévations sont en *Int16*. Vous devez utiliser `band::GetScale()` et `band::GetOffset()` pour les convertir en mètres.

B Écriture

Utilisez l'appel *Create*. Définissez l'option *MINUSERPIXELVALUE* (de type *float*) pour l'élévation la plus basse de vos données élévation, et *MAXUSERPIXELVALUE* à la plus haute. L'unité doit correspondre à l'unité de l'élévation que vous donnerez à `band::SetUnitType()`.

Appelez `dataset::SetProjection()` et `dataset::SetGeoTransform()` avec les détails du système de coordonnées. Autrement, le pilote n'encodera pas les élévations physique proprement. Les systèmes de coordonnées géographique (basé sur des degrés) seront convertie en système local basé sur des mètres.

Pour garder la précision, les meilleures hauteur de base et d'échelle seront utilisé pour utiliser au mieux le domaine sur 16 bit.

Les élévations sont en *Float32*.

C Roundtripping

Les erreurs par trip tendent à quelques centimètres pour les élévations et jusqu'à un ou deux mètres pour l'étendue au sol si des systèmes de coordonnées basé sur les degrés sont écrit. Les gros MNT en degré implique des distorsions inévitable depuis que le pilote utilise seulement des mètres.

D Historique

- **v1.0 (Mar 26/06) :** Création ;
- **v1.1 (Apr 20/06) :** Ajout des corrections de la lecture de SIZE et de la gestion de `Create()` ;
- **v1.2 (Jun 6/07) :** Amélioration de la détermination de l'échelle et de la hauteur de base lors de l'écriture.

Lisez également :

- Implémenté dans *gdal/frmts/terragen/terragendataset.cpp*.
- Lisez le fichier readme.txt pour l'installation et la gestion des informations.
- Spécification des fichiers Terragen Terrain.

Chapitre LXXII USGSDEM -- USGS ASCII DEM (et CDED)

GDAL inclut la gestion de la lecture des fichiers USGS ASCII DEM. C'est le format traditionnel utilisé par l'USGS avant d'être remplacé par le SDTS, et est le format utilisé pour les données CDED DEM du Canada. La plupart des variations populaires sur les fichiers USGS DEM doivent être géré, en incluant la reconnaissance correcte des systèmes de coordonnée et le géoréférencement. Les fichiers DEM de l'USGS de 7,5 minutes (grille UTM) ont généralement des zones de données manquante près des bords, et ceux-ci sont proprement noté avec des valeurs *nodata*. Les valeurs d'élévation dans les fichiers DEM de l'USGS peuvent être en mètre ou en pied, et cela sera indiqué par la valeur retour de `GDALRasterBand::GetUnitType()` (soit "m" soit "ft").

Notez que les fichiers DEM de l'USGS sont représentés par une seule tuile. Cela peut induire des problèmes de cache si la taille du cache des tuiles de GDAL est petite. Il en résultera également un délai substantiel lorsque le premier pixel est lu tandis que le fichier entier sera importé.

Une partie de code pour implémenter ce format est dérivé du code de VTP par Ben Discoe. Voyez le projet Virtual Terrain (http://www.vterrain.org/) pour plus d'information sur VTP.

A Problèmes de création

GDAL gère l'export des fichiers de données DEM de l'USGS et CDED en géographique (et UTM), ainsi que la possibilité de générer des produits CDED 2.0 50K selon les spécifications du gouvernement fédéral Canadien.

Les données en entrées doivent déjà être échantillonné dans un système de coordonnées UTM ou en coordonnées géographiques. Par défaut la zone entière du fichier d'entrée sera exportée, mais pour les produits CDED50K le fichier créé sera échantillonné à la résolution définie à la production et dans les limites des tuiles du produit.

Si le fichier en entré a des informations définies appropriées sur le système de coordonnée, l'export vers des formats de produit spécifique peut prend une entrée dans différents systèmes de coordonnées (c'est à dire projection Albers vers géographique NAD83 pour la production CDED 50K).

Options de création :

- **PRODUCT=DEFAULT/CDED50K :** quand CDED50K est sélectionné, le fichier de sortie sera obligé d'adhérer aux spécifications CDED 50K. La sortie aura toujours une taille de 1201x1201 et généralement une tuile de 15 minute par 15 minute (bien que plus large dans les longitudes dans les zones plus au nord).
- **TOPLEFT=long,lat :** pour les produits CDED50K, cela est utilisé pour définir le coin en haut à gauche de la tuile à générer. Il doit être sur une limite de 15 minutes et peut être donnée en degrés décimal ou en degrés et minutes (par exemple `TOPLEFT=117d15w,52d30n`).
- **RESAMPLE=Nearest/Bilinear/Cubic/CubicSpline :** définie le noyau d'échantillonnage utilisé pour échantillonner les données à la grille cible. A un effet seulement lorsque un produit particulier comme CDED50K est produit. *Bilinear* par défaut.
- **DEMLevelCode=integer :** niveau DEM (1, 2 ou 3 si définie). 1 par défaut.
- **DataSpecVersion=integer :** version/révision données et spécification (par exemple. 1020)
- **PRODUCER=text :** jusqu'à 60 caractères peuvent être placé dans le champ producteur du fichier généré.
- **OriginCode=text :** jusqu'à 4 caractères peuvent être placé dans le champ code d'origine du fichier généré (YT pour Yukon).
- **ProcessCode=code :** Un seul caractère peut être placé dans le champ code du processus du fichier généré (8=ANUDEM, 9=FME, A=TopoGrid).
- **TEMPLATE=filename :** pour n'importe quel fichier de sortie, un ficher modèle peut être définie. Un certain nombre de champ (Data Producer inclut) sera copié du fichier modèle s'il est fournit, et sera sinon laissé vide.
- **ZRESOLUTION=float :** les MNT stockent les informations d'élévation sous forme d'entier positif, et ces entiers sont échelonné en utilisant la « résolution z ». Par défaut, cette résolution est de 1.http:*0. Cependant, vous pouvez définir ici une résolution différente, si vous voulez que vos entiers soient échantillonnés dans des points flottants.
- **NTS=name :** nom du Mapsheet NTS, utilisé pour dériver TOPLEFT. A seulement un effet lorsque les produits particulier comme CDED50K ont été produit.
- **INTERNALNAME=name :** nom du jeu de données écrit dans l'en-tête du fichier. A seulement un effet quand des produits particulier comme CDED50K ont été produit.

Exemple : La commande suivante générera un tuile simple CDED50K, en extrayant d'une couverture plus large du MNT yk_3arcsec pour une tuile avec un coin en haut à gauche à -117w,60n. Le fichier *yk_template.dem* est utilisé pour définir des champs du produit incluant les champs Producteur de Données (Data Producter), Code du Processus et Code d'Origine.

```
gdal_translate -of USGSDEM -co PRODUCT=CDED50K -co
TEMPLATE=yk_template.dem \
          -co TOPLEFT=-117w,60n yk_3arcsec 031a01_e.dem
```

Note!

implémenté dans gdal/frmts/usgsdem/usgsdemdataset.cpp.

Le code de lecture des MNT de l'USGS dans GDAL est dérivé de l'importateur du logiciel VTP (http://www.vterrain.org/). Les possibilité d'export a été développé avec l'aide financière du *Yukon Department of Environment*.

Chapitre LXXIII Format virtuel de GDAL

A Introduction

Le pilote VRT est un pilote de format pour GDAL qui permet de créer des jeux de données GDAL virtuel à partir d'autres jeux de données GDAL avec des repositionnements et potentiellement des algorithmes appliqués ainsi que divers types d'ajout et de modification de méta-données. Les descriptions VRT des jeux de données peuvent être sauvé dans un format XML avec l'extension .vrt.

Un exemple d'un fichier .vrt simple se référent à un jeu de données de 512x512 avec une bande chargé à partir d'un fichier *utm.tif* ressemblerait à ceci :

```
<VRTDataset rasterXSize="512" rasterYSize="512">
<GeoTransform>440720.0, 60.0, 0.0, 3751320.0, 0.0,
-60.0</GeoTransform>
<VRTRasterBand dataType="Byte" band="1">
    <ColorInterp>Gray</ColorInterp>
    <SimpleSource>
    <SourceFilename relativeToVRT="1">utm.tif</SourceFilename>
    <SourceBand>1</SourceBand>
    <SrcRect xOff="0" yOff="0" xSize="512" ySize="512"/>
    <DstRect xOff="0" yOff="0" xSize="512" ySize="512"/>
    </SimpleSource>
</VRTRasterBand>
</VRTDataset>
```

De nombreux aspects des fichiers VRT sont la conséquence directe de l'encodage XML du modèle de données de GDAL qui devraient être revus pour la compréhension de la sémantique des différents éléments.

Les fichiers VRT peuvent être produit par traduction vers le format VRT. Le fichier résultat peut alors être édité pour modifier la cartographie, ajouter des méta-données ou d'autres choses. Les fichiers VRT peuvent aussi être produit par programmation de diverses manières.

Cette section couvrira le format de fichier .vrt (appropriée pour les utilisateurs éditant des fichiers .vrt), et comment les fichiers .vrt peuvent être crée et manipulé par

programmation pour les développeurs.

B Format .vrt

Les fichiers virtuels stockés sur le disque sont laissé au format XML avec les éléments suivants :

- **VRTDataset :** c'est l'élément racine pour l'ensemble du jeu de données GDAL. Il doit avoir les attributs rasterXSize et rasterYSize décrivant la largeur et la hauteur du jeu de données en pixels. Il peut avoir des sous-éléments SRS, GeoTransform, GCPList, Metadata, MaskBand et VRTRasterBand.

```
<VRTDataset rasterXSize="512" rasterYSize="512">
```

Les sous-éléments autorisés pour VRTDataset sont :

- **SRS :** cet élément contient le système de référence spatial (système de coordonnées) au format WKT de l'OGC. Notez qu'il doit être échappé pour le XML, ainsi les items comme les guillemets auront les séquences esperluette (&) d'échappement substitué. De même, le WKT, et les méthodes *SetFromUserInput()* valide en entrée (tel que les noms well known GEOGCS, et le format PROJ.4) est également autorisé dans l'élément SRS.

```
<SRS>PROJCS[&quot;NAD27 / UTM zone
11N&quot;,GEOGCS[&quot;NAD27&quot;,DATUM[&quot;North_American
_Datum_1927&quot;,SPHEROID[&quot;Clarke
1866&quot;,6378206.4,294.9786982139006,AUTHORITY[&quot;EPSG&q
uot;,&quot;7008&quot;]],AUTHORITY[&quot;EPSG&quot;,&quot;6267
&quot;]],PRI
MEM[&quot;Greenwich&quot;,0],UNIT[&quot;degree&quot;,0.017453
2925199433],AUTHORITY[&quot;EPSG&quot;,&quot;4267&quot;]],PRO
JECTION[&quo
t;Transverse_Mercator&quot;],PARAMETER[&quot;latitude_of_orig
in&quot;,0],PARAMETER[&quot;central_meridian&quot;,-
117],PARAMETER[&quot;
scale_factor&quot;,0.9996],PARAMETER[&quot;false_easting&quot
;,500000],PARAMETER[&quot;false_northing&quot;,0],UNIT[&quot;
metre&quot;,
1,AUTHORITY[&quot;EPSG&quot;,&quot;9001&quot;]],AUTHORITY[&qu
ot;EPSG&quot;,&quot;26711&quot;]]</SRS>
```

- **GeoTransform :** cet élément contient une transformation spatiales affine à 6 valeurs pour le jeu de données, créant une cartographie entre les coordonnées en pixel/ligne et les coordonnées géoréférencées.

```
<GeoTransform>440720.0,  60,  0.0,  3751320.0,  0.0,
-60.0</GeoTransform>
```

- **Metadata :** cet élément contient une liste de pair nom/valeur de méta-données associé à VRTDataset comme un tout, ou à VRTRasterBand. Il a un sous-élément <MDI> (méta-données item) qui possède un attribut "key" et la valeur comme une donnée de cet élément.

```
<Metadata>
    <MDI key="md_key">Metadata value</MDI>
</Metadata>
```

- **MaskBand :** (GDAL >= 1.8.0) cet élément représente une bande de masque qui est partagé entre toutes les bandes sur le jeu de données (voir GMF_PER_DATASET dans RFC 15). Il doit contenir un seul élément enfant VRTRasterBand, qui est la description de la bande du masque lui-même.

```
<MaskBand>
    <VRTRasterBand dataType="Byte">
    <SimpleSource>
        <SourceFilename
relativeToVRT="1">utm.tif</SourceFilename>
        <SourceBand>mask,1</SourceBand>
        <SrcRect xOff="0" yOff="0" xSize="512" ySize="512"/>
        <DstRect xOff="0" yOff="0" xSize="512" ySize="512"/>
    </SimpleSource>
    </VRTRasterBand>
</MaskBand>
```

- **VRTRasterBand** : il représente une bande du jeu de données. il aura un attribut dataType avec le type de données pixel associé à cette bande (utilise les termes de Byte, UInt16, Int16, UInt32, Int32, Float32, Float64, CInt16, CInt32, CFloat32 ou CFloat64) et la bande que cete élément représente (1 based). Cet élément peut avoir des sous-éléments Metadata, ColorInterp, NoDataValue, HideNoDataValue, ColorTable, Description et MaskBand ainsi que diverses éléments sources tel que SimpleSource, ComplexSource, etc. Une bande raster peut avoir plusieurs « sources » indiquant d'où les données du raster réel doivent être recherché, et comment il doit être drapé dans l'espace des pixels des bandes du raster.

Les sous-éléments autorisés pour VRTRasterBand sont :

- **ColorInterp :** la données de cet élément doit être le nom d'un type d'interprétation de couleur . Un parmi *Gray*, *Palette*, *Red*, *Green*, *Blue*, *Alpha*, *Hue*, *Saturation*, *Lightness*, *Cyan*, *Magenta*, *Yellow*, *Black*, ou *Unknown*.

```
<ColorInterp>Gray</ColorInterp>:
```

- **NoDataValue :** élément existe une bande raster a une valeur *nodata* associé à la valeur données dans cet élément.

```
<NoDataValue>-100.0</NoDataValue>
```

- **HideNoDataValue :** si cette valeur est 1, la valeur *nodata* ne sera pas renvoyée. Essentiellement, le *caller* ne sera pas au courant du pixel *nodata* quand il en lit un. Tout jeux de données copié/traduit à partir de celui-ci n'aura pas de valeur *nodata*. Ceci est utile lorsque vous voulez spécifier

une valeur d'arrière plan fixe pour le jeu de données. L'arrière plan sera la valeur définie par l'élément NoDataValue.
La valeur par défaut est 0 quand cet élément est absent.

```
<HideNoDataValue>1</HideNoDataValue>
```

- **ColorTable :** cet élément est un parent d'élément Entry définissant les entrées dans une table de couleur. Pour l'instant seul les tables de couleurs RVBA sont gérées avec c1 correspondant au rouge, c2 au vert, c3 au bleu et c4 au canal alpha. Les entrées sont ordonnées et sont présumé démarrer à l'entrée 0 de la table de couleur.

```
<ColorTable>
<Entry c1="0" c2="0" c3="0" c4="255"/>
<Entry c1="145" c2="78" c3="224" c4="255"/>
</ColorTable>
```

- **Description :** cet élément contient la description optionnelle d'une bande raster au format texte.

```
<Description>Crop Classification Layer</Description>
```

- **UnitType :** cet élément optionnel contient l'unité vertical pour les données de la bande d'élévation. Un parmi "m" pour mètres ou "ft" pour feet. Par défaut les mètres sont utilisé.

```
<UnitType>ft</UnitType>
```

- **Offset :** cet élément optionnel contient l'offset qui doit être appliqué lors du calcul des pixel réel à partir des valeurs du pixel sur une bande raster. 0.0 par défaut.

```
<Offset>0.0</Offset>
```

- **Scale :** cet élément optionnel contient l'échelle qui doit être appliqué lors du calcul des valeurs du pixel réel à partir des valeurs des pixels sur une bande raster. 1.0 est la valeur par défaut.

```
<Scale>0.0</Scale>
```

- **Overview :** cet élément optionnel décrit un niveau d'aperçu pour la bande. Il doit avoir un élément enfant *SourceFilename* et *SourceBand*. L'élément *SourceFilename* peut avoir un attribut booléen *relativeToVRT*. Plusieurs éléments peuvent être utilisé pour décrire plusieurs aperçus.

```
<Overview>
<SourceFilename
relativeToVRT="1">yellowstone_2.1.ntf.r2</SourceFilename>
```

```
<SourceBand>1</SourceBand>
</Overview>
```

- **CategoryNames :** cet élément optionnel contient une liste de sous-élément de Category avec les noms des catégories pour les bandes raster classifiées.

```
<CategoryNames>
    <Category>Missing</Category>
    <Category>Non-Crop</Category>
    <Category>Wheat</Category>
    <Category>Corn</Category>
    <Category>Soybeans</Category>
</CategoryNames>
```

- **SimpleSource :** La balise *SimpleSource* indique que les données raster doivent être lues à partir d'un jeu de données séparés, en indiquant le jeu de données, et les bandes à partir de les lire, et comment les données doivent être drapées dans ces bandes raster. La balise *SimpleSource* peut contenir les sous-éléments *SourceFilename*, *SourceBand*, *SrcRect*, et *DstRect*.
  L'élément *SrcRect* indiquera quel rectangle du fichier source indiqué doit être lu, et l'élément *DstRect* indique comment le rectangle des données sources doit être drappé dans l'espace *VRTRasterBands*.
  L'attribut *relativeToVRT* dans l'élément *SourceFilename* indique si le nom du fichier doit être interprété comme relatif au fichier .vrt (sa valeur est 1) ou non relatif au fichier .vrt (sa valeur est 0). 0 par défaut.
  Certaines caractéristiques de la bande source peuvent être définie dans la balise optionnelle *SourceProperties* pour permettre au pilote VRT de différer l'ouverture du jeu de données source jusqu'à ce qu'il ait réellement besoin de lire les données. Cela est particulièrement utile lors de la construction de VRT avec un grand nombre de jeu de données source. Les paramètres nécessaires sont les dimensions du raster, la taille des blocs et le type de données. Si la balise *SourceProperties* n'est pas présente, le jeu de données source sera ouvert en même temps que le fichier VRT lui-même.
  À partir de GDAL 1.8.0, le contenu du sous-élément *SourceBand* peut se référer à une bande de masque. Par exemple *mask,1* signifie la bande de masque de la première bande de la source

```
<SimpleSource>
<SourceFilename relativeToVRT="1">utm.tif</SourceFilename>
<SourceBand>1</SourceBand>
<SourceProperties RasterXSize="512" RasterYSize="512"
DataType="Byte" BlockXSize="128" BlockYSize="128"/>
<SrcRect xOff="0" yOff="0" xSize="512" ySize="512"/>
```

```
<DstRect xOff="0" yOff="0" xSize="512" ySize="512"/>
</SimpleSource>
```

- **AveragedSource :** *AveragedSource* est dérivé de *SimpleSource* et partage les mêmes propriétés sauf qu'il utilise un réechentillonnage moyen au lieu de l'algorithme de plus proche voisin comme dans *SimpleSource*, quand la taille du rectangle de destination n'est pas le même que la taille du rectangle source.
- **ComplexSource :** le paramètre *ComplexSource* est dérivé de *SimpleSource* (il partage donc les éléments *SourceFilename*, *SourceBand*, *SrcRect* et *DestRect*), mais il fournit la gestion du reéchentillonage et l'écart des valeurs source. Certaines zones de la source peuvent être masquées en définissant la valeur *NODATA*.
  Le paramètre *ComplexSource* gère l'ajout de table lookup (LUK) personnalisée pour transformer les valeurs sources vers la destination. Les LUT peuvent être définie en utilisant la forme suivante :

```
<LUT>[src valeur 1]:[dest valeur 1],[src valeur 2]:[dest
valeur 2],...</LUT>
```

  Les valeurs intermédiaire sont calculées en utilisant une interpolation linéaire entre les valeurs de destination de liaison du domaine correspondant.
  Le paramètre *ComplexSource* gère la recherche de composant de couleur d'une bande raster source qui possède une table de couleur. La valeur *ColorTableComponent* est l'index du composant de couleur à extraire : 1 pour la bande rouge, 2 pour la bande verte, 3 pour la bande bleue ou 4 pour la bande alpha.
  Lors de la transformation des valeurs sources les opérations sont exécutées dans l'ordre suivant :
  1. masquage des Nodata ;
  2. expansion de la table de couleur ;
  3. application du ratio d'échelle ;
  4. application du décalage d'échelle ;
  5. lecture de la table.

```
<ComplexSource>
<SourceFilename relativeToVRT="1">utm.tif</SourceFilename>
<SourceBand>1</SourceBand>
<ScaleOffset>0</ScaleOffset>
<ScaleRatio>1</ScaleRatio>
<ColorTableComponent>1</ColorTableComponent>
<LUT>0:0,2345.12:64,56789.5:128,2364753.02:255</LUT>
<NODATA>0</NODATA>
<SrcRect xOff="0" yOff="0" xSize="512" ySize="512"/>
<DstRect xOff="0" yOff="0" xSize="512" ySize="512"/>
</ComplexSource>
```

- **KernelFilteredSource :** c'est un pixel source dérivé de

Simple Source (il partage donc les éléments SourceFilename, SourceBand, SrcRect et DestRect éléments), mais il passe également les données à travers un simple filtre définie avec l'élément Kernel. L'élément Kernel doit avoir deux éléments enfants, Size et Coefs et en option l'attribut booléen normalisé (par défaut à false=0). La taille doit doit toujours être un nombre impair, et la paramètre Coefs doit contenir Size * Size entrées séparées par des espaces.

```
<KernelFilteredSource>
<SourceFilename>/debian/home/warmerda/openev/utm.tif</SourceF
ilename>
<SourceBand>1</SourceBand>
<Kernel normalized="1">
    <Size>3</Size>
    <Coefs>0.11111111 0.11111111 0.11111111 0.11111111
0.11111111 0.11111111 0.11111111 0.11111111
0.11111111</Coefs>
</Kernel>
</KernelFilteredSource>
```

- **MaskBand :** (GDAL >= 1.8.0) cet élément représente une bande de masque qui est spécifique à *VRTRasterBand* qu'il contient. Il doit contenir un seule élément enfant *VRTRasterBand*, qui est la description de la bande de masque lui-même.

C Description des .vrt pour les fichiers brutes

Jusqu'ici nous avons décris comment dérivé de nouveaux jeux de données à partir de fichiers existants géré par GDAL. Cependant, il est également commun d'avoir à utiliser des fichiers raster binaires brutes pour lesquels la structure des données est connus mais pour lequel aucun pilote spécifique à ce format n'existe. Cela peut être accomplit en écrivant un fichier .vrt décrivant le fichier brute.

Par exemple, le fichier .vrt suivant décrit un fichier raster brute contenant des pixels complexes en point flottant dans un fichier appelé *l2p3hhsso.img*. Les données images débutent à partir du premier byte (`mageOffset=0`). La distance des bytes entre les pixels est de 8 (`PixelOffset=8`), la taille d'un *Cfloat32*. La distance es bytes du début d'une ligne au début de la suivante est de 9376 bytes (`LineOffset=9376`) ce qui correspond à la largeur (1172) fois la taille d'un pixel (8).

```
<VRTDataset rasterXSize="1172" rasterYSize="1864">
    <VRTRasterBand dataType="CFloat32" band="1"
subClass="VRTRawRasterBand">
        <SourceFilename
relativetoVRT="1">l2p3hhsso.img</SourceFilename>
        <ImageOffset>0</ImageOffset>
        <PixelOffset>8</PixelOffset>
        <LineOffset>9376</LineOffset>
        <ByteOrder>MSB</ByteOrder>
    </VRTRasterBand>
</VRTDataset>
```

Il est à noter que VRTRasterBand a un déterminant subClass de "VRTRawRasterBand". Également, VRTRawRasterBand contient un nombre d'éléments non vue précédemment mais aucune information « source ». VRTRawRasterBands peut ne jamais avoir de sources (c'est à dire SimpleSource), mais doit contenir les éléments suivants en plus de tous les éléments de méta-données précédemment décrit qui sont encore géré.

- `SourceFilename` : le nom du ficher brute contenant les données pour cette bande. L'attribut relativeToVRT peut être utilisé pour indiquer si SourceFilename est relative au fichier .vrt (1) ou non (0).
- `ImageOffset` : la distance en bytes du début du premier pixel de données de cette bande d'image. Zéro par défaut.
- `PixelOffset` : la distance en bytes du début d'un pixel et du suivant sur la même ligne. Dans des données simples en paquet (packed single band) cela correspondra à la taille de dataType en bytes.
- `LineOffset` : la distance en bytes du début de la ligne de données et de la suivante. Dans les données simple en paquet (packed single band) cela correspondra à PixelOffset * rasterXSize.
- `ByteOrder` : définie l'ordre des bytes des données sur le disque. Soit LSB (*Least Significant Byte first*) tel que l'ordre naturel sur les systèmes Intel x86 systems ou MSB (*Most Significant Byte first*) tel que sur les systèmes Motorola ou Sparc systems. Par défaut celui de l'ordre de la machine locale.

D'autre remarques :

Les données de l'image sur le disque sont supposées être du même type de données que la bande dataType de *VRTRawRasterBand*. Tous les attributs ne venant pas de la source du *VRTRasterBand* sont gérés, incluant les tables de couleurs, les méta-données, et l'interprétation des couleurs.

VRTRawRasterBand gère la mise à jour du raster alors que la source basé sur *VRTRasterBand* est toujours en lecture seule. L'outil OpenEV inclut un menu Fichier pour entrer des paramètres décrivant le fichier raster brute dans nue interface graphique et créer le fichier .vrt correspondant.

Les bandes multiples dans un fichier .vrt peuvent venir du même fichier brute. Assurez vous juste que les définitions *ImageOffset*, *PixelOffset*, et *LineOffset* pour chaque bande sont appropriées pour le pixel de cette bande particulière. Un autre exemple, dans ce cas une image de pixel entrelacé de 400x300 RVB.

```
<VRTDataset rasterXSize="400" rasterYSize="300">
    <VRTRasterBand dataType="Byte" band="1"
subClass="VRTRawRasterBand">
        <ColorInterp>Red</ColorInterp>
        <SourceFilename relativetoVRT="1">rgb.raw</SourceFilename>
        <ImageOffset>0</ImageOffset>
        <PixelOffset>3</PixelOffset>
        <LineOffset>1200</LineOffset>
    </VRTRasterBand>
    <VRTRasterBand dataType="Byte" band="2"
subClass="VRTRawRasterBand">
        <ColorInterp>Green</ColorInterp>
        <SourceFilename relativetoVRT="1">rgb.raw</SourceFilename>
        <ImageOffset>1</ImageOffset>
        <PixelOffset>3</PixelOffset>
        <LineOffset>1200</LineOffset>
    </VRTRasterBand>
```

```
    <VRTRasterBand dataType="Byte" band="3"
subClass="VRTRawRasterBand">
        <ColorInterp>Blue</ColorInterp>
        <SourceFilename relativetoVRT="1">rgb.raw</SourceFilename>
        <ImageOffset>2</ImageOffset>
        <PixelOffset>3</PixelOffset>
        <LineOffset>1200</LineOffset>
    </VRTRasterBand>
</VRTDataset>
```

D Création programmée de jeux de données VRT

Le pilote VRT gère plusieurs méthodes de création de jeux de données VRT. En tant que partie de GDAL 1.2.0 le fichier inclue *vrtdataset.h* doit être installé avec les fichiers inclues coeur de GDAL, permettant un accès direct au fichier aux classes VRT. Cependant, même sans cela, la plupart des possibilités resteront disponible à travers les interfaces standards de GDAL.

Pour créer un jeu de données VRT qui est un clone d'un jeu de données existants utilisez la méthode *CreateCopy()*. Par exemple pour cloner utm.tif dans un fichier wrk.vrt en C++ le code suivant pourra être utilisé :

```
GDALDriver *poDriver = (GDALDriver *) GDALGetDriverByName( "VRT" );
GDALDataset *poSrcDS, *poVRTDS;

poSrcDS = (GDALDataset *) GDALOpenShared( "utm.tif", GA_ReadOnly );

poVRTDS = poDriver->CreateCopy( "wrk.vrt", poSrcDS, FALSE, NULL, NULL,
NULL );

GDALClose((GDALDatasetH) poVRTDS);
GDALClose((GDALDatasetH) poSrcDS);
```

Notez l'utilisation de *GDALOpenShared()* lors de l'ouverture du jeu de données source. Il est conseillé d'utiliser *GDALOpenShared()* dans cette situation afin d'être capable de publier la référence explicite à celle-ci avant de fermer le jeu de données VRT lui-même. En d'autes mots, dans l'exemple précédent, vous pouvez également inverser les deux dernières lignes, tandis que si vous ouvrez le jeu de données source avec *GDALOpen()*, vous devrez fermer le jeu de données VRT avant de fermer le jeu de données source.

Pour créer une copie virtuelle d'un jeu de données avec des attributs ajoutés ou modifiés tels que les méta-données ou les systèmes de coordonnées qui sont souvent difficile de changer dans les autres formats, vous pouvez faire ce qui suit. Dans ce cas, le jeu de données virtuel est crée « en mémoire » seulement par virtualisation de sa création avec un nom de fichier vide, puis utilisé comme source modifiée pour passer à une méthode *CreateCopy()* créant le format TIFF.

```
poVRTDS = poDriver->CreateCopy( "", poSrcDS, FALSE, NULL, NULL,
NULL );

poVRTDS->SetMetadataItem( "SourceAgency", "United States Geological
Survey");
poVRTDS->SetMetadataItem( "SourceDate", "July 21, 2003" );
```

```
poVRTDS->GetRasterBand( 1 )->SetNoDataValue( -999.0 );

GDALDriver *poTIFFDriver = (GDALDriver *) GDALGetDriverByName( "GTiff"
);
GDALDataset *poTiffDS;

poTiffDS = poTIFFDriver->CreateCopy( "wrk.tif", poVRTDS, FALSE, NULL,
NULL, NULL );

GDALClose((GDALDatasetH) poTiffDS);
```

Dans les exemples ci-dessus la valeur *nodata* est définie à -999. Vous pouvez définir l'élément *HideNoDataValue* dans la bande du jeu de données VRT en utilisant *SetMetadataItem()* sur cette bande.

```
poVRTDS->GetRasterBand( 1 )->SetMetadataItem( "HideNoDataValue" ,
"1" );
```

Dans cet exemple, un jeu de données est crée avec la méthode Create(), et on ajoute des bandes et des sources par programmation, mais toujours à l'aide de l'API « générique ». Un attribut spécial des jeux de données VRT permet d'ajouter des sources aux VRTRasterBand (mais pas à VRTRawRasterBand) en passant le XML décrivant la source dans *SetMetada()* sur la cible du domaine spécial « new_vrt_sources ». Le domaine cible « vrt_sources » peut également être utilisé, auquel cas n'importe quelle source sera rejetée avant d'en ajouter de nouvelle. Dans cet exemple nous construisons un simple filtre moyen à la place de source simple.

```
// construct XML for simple 3x3 average filter kernel source.
const char *pszFilterSourceXML  =
"<KernelFilteredSource>"
"  <SourceFilename>utm.tif</SourceFilename><SourceBand>1</SourceBand>"
"  <Kernel>"
"    <Size>3</Size>"
"    <Coefs>0.111 0.111 0.111 0.111 0.111 0.111 0.111 0.111
0.111</Coefs>"
"  </Kernel>"
"</KernelFilteredSource>";

// Create the virtual dataset.
poVRTDS = poDriver->Create( "", 512, 512, 1, GDT_Byte, NULL );
poVRTDS->GetRasterBand(1)-
>SetMetadataItem("source_0",pszFilterSourceXML",
                                          "new_vrt_sources");
```

Une manière plus générale de cela et qui produira un clone 3x3 moyen de n'importe quelle source de données en entrée pourrait ressembler à ce qui suit. Dans ce cas nous définissons délibérément la source de données filtrée comme dans le domaine « vrt_sources » pour écraser la SimpleSource crée par la méthode *CreateCopy()*. Le fait que nous utilisons CreateCopy() nous assure que tous les autres méta-données, géoréférencement et autre seront préservé à partir du jeu de données source ... La seule chose que nous somme en train de changer est la source des données pour chaque bande.

```
int   nBand;
GDALDriver *poDriver = (GDALDriver *) GDALGetDriverByName( "VRT" );
GDALDataset *poSrcDS, *poVRTDS;

poSrcDS = (GDALDataset *) GDALOpenShared( pszSourceFilename,
GA_ReadOnly );

poVRTDS = poDriver->CreateCopy( "", poSrcDS, FALSE, NULL, NULL,
NULL );

for( nBand = 1; nBand <= poVRTDS->GetRasterCount(); nBand++ )
{
    char szFilterSourceXML[10000];

    GDALRasterBand *poBand = poVRTDS->GetRasterBand( nBand );

    sprintf( szFilterSourceXML,
        "<KernelFilteredSource>"
        "  <SourceFilename>%s</SourceFilename><SourceBand>
%d</SourceBand>"
        "  <Kernel>"
        "    <Size>3</Size>"
        "    <Coefs>0.111 0.111 0.111 0.111 0.111 0.111 0.111 0.111
0.111</Coefs>"
        "  </Kernel>"
        "</KernelFilteredSource>",
        pszSourceFilename, nBand );

    poBand->SetMetadataItem( "source_0", szFilterSourceXML,
"vrt_sources" );
}
```

La classe *VRTDataset* est une des quelques implémentations de jeux de données qui gère la méthode *AddBand()*. Les options passées à la méthode *AddBand()* peut être utilisées pour contrôler le type de bande créé (*VRTRasterBand*, *VRTRawRasterBand*, *VRTDerivedRasterBand*), et dans le cas de *VRTRawRasterBand* de définir ses différentes paramètres. Pour le standard *VRTRasterBand*, les sources doivent être définie avec les exemples *SetMetadata()* / *SetMetadataItem()* ci-dessus.

```
GDALDriver *poDriver = (GDALDriver *) GDALGetDriverByName( "VRT" );
GDALDataset *poVRTDS;

poVRTDS = poDriver->Create( "out.vrt", 512, 512, 0, GDT_Byte, NULL );
char** papszOptions = NULL;
papszOptions = CSLAddNameValue(papszOptions, "subclass",
"VRTRawRasterBand"); // if not specified, default to VRTRasterBand
papszOptions = CSLAddNameValue(papszOptions, "SourceFilename",
"src.tif"); // mandatory
papszOptions = CSLAddNameValue(papszOptions, "ImageOffset", "156"); //
optionnal. default = 0
papszOptions = CSLAddNameValue(papszOptions, "PixelOffset", "2"); //
optionnal. default = size of band type
papszOptions = CSLAddNameValue(papszOptions, "LineOffset", "1024"); //
optionnal. default = size of band type * width
papszOptions = CSLAddNameValue(papszOptions, "ByteOrder", "LSB"); //
```

```
optionnal. default = machine order
papszOptions = CSLAddNameValue(papszOptions, "RelativeToVRT", "true");
// optionnal. default = false
poVRTDS->AddBand(GDT_Byte, papszOptions);
CSLDestroy(papszOptions);

delete poVRTDS;
```

E Utilisation des bandes dérivées

Un type de bande spécialisé est une bande 'dérivée' qui dérive ses informations des pixels de ses bandes sources. Avec ce type de bande vous devez définir une fonction pixel, qui a la responsabilité de générer le raster de sortie. Les fonctions pixel sont crée par une application puis enregistré avec GDAL en utilisant une clé unique.

En utilisant des bandes dérivées vous pouvez créer des jeux de données VRT qui manipule des bandes à la volées sans créer de nouveau fichier de bandes sur le disque. Par exemple, vous pouvez générer une bande en utilisant 4 bandes source à partir d'une neuvième bande d'un jeu de données en entré (x0, x3, x4, et x8) : band_value = sqrt((x3*x3+x4*x4)/(x0*x8));

Vous pouvez écrire la fonction pixel pour calculer cette valeur puis l'enregistrer avec GDAL avec le nom « MyPremiereFonction ». Puis, le fichier XML VRT suivant pourra être utilisé pour afficher cette bande dérivée :

```
<VRTDataset rasterXSize="1000" rasterYSize="1000">
    <VRTRasterBand dataType="Float32" band="1"
subClass="VRTDerivedRasterBand">>
        <Description>Magnitude</Description>
        <PixelFunctionType>MyFirstFunction</PixelFunctionType>
        <SimpleSource>
            <SourceFilename
relativeToVRT="1">nine_band.dat</SourceFilename>
            <SourceBand>1</SourceBand>
            <SrcRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
            <DstRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
        </SimpleSource>
        <SimpleSource>
            <SourceFilename
relativeToVRT="1">nine_band.dat</SourceFilename>
            <SourceBand>4</SourceBand>
            <SrcRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
            <DstRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
        </SimpleSource>
            <SimpleSource>
            <SourceFilename
relativeToVRT="1">nine_band.dat</SourceFilename>
            <SourceBand>5</SourceBand>
            <SrcRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
            <DstRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
        </SimpleSource>
        <SimpleSource>
            <SourceFilename
relativeToVRT="1">nine_band.dat</SourceFilename>
```

```
            <SourceBand>9</SourceBand>
            <SrcRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
            <DstRect xOff="0" yOff="0" xSize="1000" ySize="1000"/>
        </SimpleSource>
    </VRTRasterBand>
</VRTDataset>
```

En plus de la spécification de la sous-classe (VRTDerivedRasterBand) et la valeur de PixelFunctionType, il y a un nouveau paramètre qui peut être utile : sourceTransferType. Typiquement, les rasters sources sont obtenu en utilisant le type de donnée de la bande dérivée. Parfois, il se peut que lorsque vous voulez que la fonction pixel puisse accéder à une source de données de plus haute résolution que le type de donnée qui est générée. Par exemple, vous pouvez avoir une bande dérivée de type « FLOAT », qui prend une source simple de type « CFloat32 » ou « CFloat64 » et renvoi la portion imaginaire. Pour accomplir cela, définissez le paramètre SourceTransfertType à « CFloat64 ». Autrement la source sera converti en « Float » avant d'appeler la fonction pixel, et la partie imaginaire sera perdue.

```
<VRTDataset rasterXSize="1000" rasterYSize="1000">
    <VRTRasterBand dataType="Float32" band="1"
subClass="VRTDerivedRasterBand">>
        <Description>Magnitude</Description>
        <PixelFunctionType>MyFirstFunction</PixelFunctionType>
        <SourceTransferType>CFloat64</SourceTransferType>
        ...
```

F Écrire des fonctions pixels

Pour enregistrer cette fonction avec GDAL (avant d'accéder à un jeu de données VRT avec des bandes dérivées qui utilisent cette fonction), une application appelle `GDALAddDerivedBandPixelFunc` avec une clé et `GDALDerivedPixelFunc` :
`GDALAddDerivedBandPixelFunc("MyFirstFunction", TestFunction);`

Le bon moment pour faire cela se situe au début d'une application quand les pilotes GDAL sont enregistrés. GDALDerivedPixelFunc est définie avec une signature similaire à IRasterIO :

Paramètres :

papoSources	Un pointeur pour entasser des rasters ; un par source. Tous leurs types de donnée doivent être le même, définie dans le paramètre eSrcType.
nSources	Le nombre de source rasters.
pData	Le buffer dans lequel les données doivent être lu, ou dans lequel il doit être écrit. Ce buffer doit contenir au moins nBufXSize * nBufYSize mots de type eBufType. L'ordre des pixel est organisé de gauche à droite, de haut en bas. L'espacement est contrôlé par les paramètres nPixelSpace, et nLineSpace.
nBufXSize	La largeur du buffer de l'image dans laquelle la région désirée doit être lue, ou dans lequel il doit être écrit.
nBufYSize	La hauteur du buffer de l'image dans laquelle la région désirée doit être

	lue, ou dans lequel il doit être écrit.
eSrcType	Le type des valeurs des pixels dans le tableau raster papoSources.
eBufType	Le type des valeurs des pixels que la fonction pixel doit générer dans le buffer de données pData.
nPixelSpace	La distance des bytes du début de la valeur d'un pixel dans pData de la prochaine valeur du pixel dans une ligne. Si la valeur par défaut doit être utilisée la taille du type de données eBufType est utilisée.
nLineSpace	La distance des bytes à partir du début d'une ligne dans pData au début de la suivante.

Retour :

```
CE_Failure on failure, otherwise CE_None.
typedef CPLErr
(*GDALDerivedPixelFunc)(void **papoSources, int nSources, void *pData,
                    int nXSize, int nYSize,
                    GDALDataType eSrcType, GDALDataType eBufType,
                    int nPixelSpace, int nLineSpace);
```

Ce qui suit est une implémentation de la fonction pixel :

```
#include "gdal.h"

CPLErr TestFunction(void **papoSources, int nSources, void *pData,
                int nXSize, int nYSize,
                GDALDataType eSrcType, GDALDataType eBufType,
                int nPixelSpace, int nLineSpace)
{

    int ii, iLine, iCol;
    double pix_val;
    double x0, x3, x4, x8;

    // ---- Init ----
    if (nSources != 4) return CE_Failure;

    // ---- Set pixels ----
    for( iLine = 0; iLine < nYSize; iLine++ )
    {
        for( iCol = 0; iCol < nXSize; iCol++ )
        {
            ii = iLine * nXSize + iCol;

            /* Source raster pixels may be obtained with SRCVAL macro
*/
            x0 = SRCVAL(papoSources[0], eSrcType, ii);
            x3 = SRCVAL(papoSources[1], eSrcType, ii);
            x4 = SRCVAL(papoSources[2], eSrcType, ii);
            x8 = SRCVAL(papoSources[3], eSrcType, ii);

            pix_val = sqrt((x3*x3+x4*x4)/(x0*x8));
```

```
            GDALCopyWords(&pix_val, GDT_Float64, 0,
                          ((GByte *)pData) + nLineSpace * iLine + iCol *
nPixelSpace,
                          eBufType, nPixelSpace, 1);
          }
    }

    // ---- Return success ---- //
    return CE_None;
}
```

G Problèmes de Multi-threading

Lors de l'utilisation de jeux de données VRT dans un environnement multi-threading, vous devez être prudent lors de l'ouverture de jeu de données VRT par le thread qui va l'utiliser par la suite. La raison de cela est que le jeu de données VRT utilise *GDALOpenShared* lors de l'ouverture des jeux de données sous-jacent. Si vous ouvrez deux fois le même jeu de données VRT par le même thread, l'ensemble des jeux de données VRT partageront la même prise en charge des jeux de données sous-jacent.

Chapitre LXXIV WCS -- OGC Web Coverage Service

Le pilote optionnel WCS pour GDAL permet l'utilisation de couverture dans un serveur WCS comme un jeu de données raster. GDAL agit comme un client au serveur WCS.

L'accès à un serveur WCS est réalisé en créant un fichier xml de description du service local ressemblant à quelque chose comme ce qui suit, avec l'url du serveur de couverture, et le nom de la couverture à accéder. Il est important qu'il n'y est pas d'espace ou d'autre caractère superflux contenu dans l'élément <WCS_GDAL>.

```
<WCS_GDAL>
    <ServiceURL>http://laits.gmu.edu/cgi-bin/NWGISS/NWGISS?
</ServiceURL>
    <CoverageName>AUTUMN.hdf</CoverageName>
</WCS_GDAL>
```

Lors de la première ouverture, GDAL cherchera la description de la couverture, et une petite image de test pour établir les détails du raster. Cette information sera mis en cache dans le fichier de description du service pour permettre une ouverture future plus rapide – aucun accès au serveur ne doit être requis tant que l'imagerie est lu pour de future ouverture.

Le pilote WCS doit gérer les serveurs WCS 1.0.0 et 1.1.0, par contre les serveurs WCS 0.7 ne sont pas gérés. N'importe quel format qui est un fichier simple et qui est dans un format géré par GDAL doit fonctionner. Le pilote préférera un format avec l'extension tiff dans le nom, sinon il se rabattra au premier format offert. Les systèmes de coordonnée sont lu à partir du résultat de *DescribeCoverage*, et doivent être sous la forme EPSG:n dans l'élément <supportedCRSs>.

Le fichier de description de service possède les éléments suivants additionnels comme enfant immédiat de l'élément *WCS_GDAL* pour être définie en option.

- **PreferredFormat :** le format à utiliser pour les appels *GetCoverage*.
- **BandCount :** nombre de bande dans le jeu de données, normalement capturé à partir de la requête de test.
- **BandType :** le type de donnée du pixel. Normalement établit à partir de la requête de test.

- **BlockXSize :** la largeur du bloc à utiliser pour l'accès distant en cache du bloc.
- **BlockYSize :** la hauteur du bloc à utiliser pour l'accès distant en cache du bloc.
- **NoDataValue :** La valeur nodata à utiliser pour toutes les bandes (vide pour pas de valeur). Normalement par défaut celui récupéré des informations de *CoverageOffering*.
- **Timeout :** le timeout à utiliser pour les requêtes au service distant. S'il n'est pas fournit, la valeur par défaut de libcurl est utilisé.
- **UserPwd :** peut être définir avec *userid:password* pour envoyer un identifiant d'utilisateur et un mot de passe au serveur distant.
- **HttpAuth :** peut être BASIC, NTLM ou ANY pour contrôler la méthode d'authentification à utiliser.
- **OverviewCount :** le nombre d'aperçu pour représenter les bandes. Par défaut un nombre tel que le haut de l'aperçu soit assez petit (plus petit que 1K x 1K environ).
- **GetCoverageExtra :** un jeu additionnel de mots-clé à ajouter aux requêtes *GetCoverage* sous la forme d'url encodé par exemple `&RESAMPLE=BILINEAR&Band=1`
- **DescribeCoverageExtra :** un jeu additionnel de mots-clé à ajouter aux requêtes *DescribeCoverage* sous la forme d'url encodé par exemple `&CustNo=775`
- **Version :** définie une version spécifique du WCS à utiliser. Pour l'instant par défaut à 1.0.0 et 1.1.0 est également géré.
- **FieldName :** nom du champ qui est accédé. Utilisé seulement avec WCS 1.1.0 et supérieur. Par défaut au premier champs du résultat de DescribeCoverage.
- **DefaultTime :** un timePosition à utiliser par défaut lors de l'accès aux couvertures avec une dimension temps. Rempli avec la dernière position du temps offerte par défaut.

A Time

À partir de GDAL 1.9.0, ce pilote inclus la gestion expérimentale des serveurs WCS 1.0.0 basés sur le temps. Lors de l'accès initial la dernière position du temps offerte sera identifié comme *DefaultTime*. Chaque position de temps disponible pour la couverture sera traité comme sous jeu de données.

Notez que les jeux de données basés sur le temps ne sont pas gérés lorsque la description du service est le nom du fichier. Pour le moment la gestion du temps n'est pas disponible pour les versions autres que WCS 1.0.0.

B Voir également

- Standards WCS de l'OGC : http://www.opengeospatial.org/standards/wcs

Chapitre LXXV WEBP - WEBP

À partir de GDAL 1.9.0, GDAL peut lire et écrire des images WebP via la bibliothèque WebP.

WebP est un nouveau format d'image qui fournit une compression sans perte pour les images photographiques. Un fichier WebP consiste de données image VP8 et d'un conteneur basé sur RIFF.

Le pilote repose sur la bibliothèque Open Source WebP (licence BSD). La bibliothèque WebP (du moins dans sa version 0.1) ne propose que la compression et la décompression des images complètes, La RAM peut donc être une limitation lorsqu'il s'agit de grandes images.

Le pilote WEBP ne gère que 3 bandes (RGB) d'images.

Le pilote WEBP peut être utilisé comme format interne utilisé par le pilote :ref:`gdal.gdal.formats.rasterlite`.

Unknown interpreted text role "ref".

A Options de création

- **QUALITY=n** par défaut l'option *quality* est définie à 75, mais cette option peut être utilisé pour sélectionner d'autres valeurs. Les valeurs doivent être comprises entre 1 et 100. Les valeurs faibles résultent en un plus grand taux de compression, mais une moins bonne qualité d'image.

B Voir également

- Home page WebP

Chapitre LXXVI WMS -- Web Map Services

Accéder à plusieurs différents types de service d'image est possible en utilisant le format WMS dans GDAL. Les services sont accédé en créant un fichier XML de description de service local - il y a des exemples ci-dessous pour chacun des services d'image gérée.

<GDAL_WMS>	
<Service name="WMS">	Définie quel mini-pilote utiliser, géré actuellement : WMS, WorldWind, TileService, TMS, TiledWMS or VirtualEarth. (nécessaire)
<Version>1.1.1</Version>	Version du WMS. (optionnel, par défaut à 1.1.1)
<ServerURL>http://onearth.jpl.nasa.gov/ wms.cgi?</ServerUrl>	URL du serveur WMS (nécessaire)
<SRS>EPSG:4326</SRS>	Projection de l'image (optionnel, par défaut à EPSG:4326, WMS version 1.1.1 ou inférieur seulement)
<CRS>CRS:83</CRS>	Projection de l'image (optionnel, défaut à EPSG:4326, WMS version 1.3.0 ou supérieur seulement)
<ImageFormat>image/jpeg</ImageFormat>	Format dans lequel on doit demander les données. Les formats avec des palettes comme image/gif seront converti en RVB. (optionnel, défaut à image/jpeg)
<Transparent>FALSE</Transparent>	Définie à TRUE pour inclure "transparent=TRUE" dans la requête GetMap du WMS (optionnel FALSE par défaut). Le format de requête et BandsCount nécessite la gestion de l'alpha.
<Layers>modis,global_mosaic</Layers>	Liste de couches séparés par des virgules (nécessaire sauf pour TiledWMS).
<TiledGroupName>Clementine</TiledGroupName>	Liste de couches séparée par une virgule (nécessaire pour TiledWMS).

<Styles></Styles>	Liste de styles séparés par des virgules. (optionnel)
<BBoxOrder>xyXY</BBoxOrder>	Ordonne les coordonnées de la bbox arbitrairement. Peut être nécessaire pour les serveurs à la version 1.3. (optionnel) x – coordonnée X basse, y – coordonnée Y basse, X - coordonnée X haute, Y - coordonnée Y haute
</Service>	
<DataWindow>	Définie la taille et l'étendue des données. (nécessaire sauf pour TiledWMS et VirtualEarth)
<UpperLeftX>-180.0</UpperLeftX>	X (longitude) coordonnée du coin haut-gauche. (optionnel, défaut à -180.0, sauf pour VirtualEarth)
<UpperLeftY>90.0</UpperLeftY>	Y (latitude) coordonnée du coin haut -gauche. (optionnel, défaut à 90.0, sauf pour VirtualEarth)
<LowerRightX>180.0</LowerRightX>	X (longitude) coordonnée du coin bas-droit. (optionnel, défaut à 180.0, sauf pour VirtualEarth)
<LowerRightY>-90.0</LowerRightY>	Y (latitude) coordonnée du coin bas-droit (optionnel, défaut à -90.0, sauf pour VirtualEarth)
<SizeX>2666666</SizeX>	Taille de l'image en pixels.
<SizeY>1333333</SizeY>	Taille de l'image en pixels.
<TileX>0</TileX>	Ajouter aux tuiles une valeur X à plus haute résolution. (ignoré pour WMS, source d'image tuilée seulement, optionnel, défaut à 0)
<TileY>0</TileY>	Ajouter aux tuiles une valeur Y à plus haute résolution. (ignoré pour WMS, source d'image tuilée seulement, optionnel, défaut à 0)
<TileLevel>0</TileLevel>	Niveau des tuiles à la plus haute résolution. (ignoré pour WMS, source d'image tuilée seulement, optionnel, défaut à 0)
<TileCountX>0</TileCountX>	Peut être utilisé pour définir la taille de l'image, SizeX = TileCountX * BlockSizeX * 2TileLevel. (source d'image tuilée seulement, optionnel, défaut à 0)
<TileCountY>0</TileCountY>	Peut être utilisé pour définir la taille de l'image, SizeY = TileCountY * BlockSizeY * 2TileLevel (source d'image tuilée seulement, optionnel, défaut à 0).
<YOrigin>top</YOrigin>	Peut être utilisé pour définir la position de l'origine Y en fonction de la grille de la tuile. Les valeurs possibles sont 'top', 'bottom', et 'default', pour lesquels le comportement est spécifique au mini-pilote (seulement le mini -pilote TMS, optionnel, 'bottom' par défaut pour TMS).

</DataWindow>	
<Projection>EPSG:4326</Projection>	Projection de l'image (optionnel, défaut à la valeur rapporté par le mini- pilote ou EPSG:4326)
<BandsCount>3</BandsCount>	Nombre de bandes/canaux, 1 pour des données en nuance de gris, 3 pour RVB. (optionnel, défaut à 3)
<BlockSizeX>1024</BlockSizeX>	Taille du bloc en pixels. (optionnel, défaut à 1024, sauf pour VirtualEarth)
<BlockSizeY>1024</BlockSizeY>	Taille du bloc en pixels. (optionnel, défaut à 1024, sauf pour VirtualEarth)
<OverviewCount>10</OverviewCount>	compte des couches à résolution réduite chacune ayant des résolutions deux fois plus petites. (optionnel, la valeur par défaut est calculé lors du lancement)
<Cache>	Active le cache disque local. Permet les opérations offline. (optionnel, par défaut sans cache)
<Path>./gdalwmscache</Path>	Endroit où stocker les fichiers du cache. Il est sain d'utiliser le même chemin de cache pour différentes sources de données. (optionnel, défaut à ./gdalwmscache)
<Depth>2</Depth>	Nombre de répertoire de couches. 2 résultera à des fichiers écrit sous la forme cache_path/A/B/ABCDEF... (optionnel, défauts à 2)
<Extension>.jpg</Extension>	Ajout aux fichiers de cache. (optionnel, défaut à none)
</Cache>	
<MaxConnections>2</MaxConnections>	nombre maximal de connections simultanée. (optionnel, 2 par défaut)
<Timeout>300</Timeout>	timeout de la connection en secondes (optionnel, 300 par défauts).
<OfflineMode>true</OfflineMode>	Ne télécharge aucune nouvelles images, utilise seulement celle du cache. Utile seulement quand le cache est activé (optionnel, défaut à *false*).
<AdviseRead>true</AdviseRead>	Active l'appel à l'API AdviseRead – télécharge les images dans le cache (optionnel, défaut à *false*).
<VerifyAdviseRead>true</VerifyAdviseRead>	Ouvre chaque image et fait des opérations de vérifications basiques avant d'écrire dans le cache. La désactivation peut avoir des cycles de CPU si le serveur est reconnus comme toujours renvoyer des images correctes. (optionnel, défaut à *true*)
<VerifyAdviseRead>true</VerifyAdviseRead>	ouvre chaque image téléchargée et réalise des vérifications basiques avant l'écriture dans le cache.

	Désactivé, cela peut sauver des cycles CPU si le serveur est reconnu comme retournant toujours des images correctes (optionnel, *true* par défaut).
<ClampRequests>false</ClampRequests>	Est ce que la requête, qui autrement serait partiellement en dehors de la fenêtre de données définie, être découpé résultant en une image plus petite que la taille du bloc demandée (optionnel, *true* par défaut).
<UserAgent>GDAL WMS driver (http://www.gdal.org/frmt_wms.html) </UserAgent>	Chaîne User-agent HTTP. Certains serveurs peuvent nécessiter un user-agent connus tel que "Mozilla/5.0" (optionnel, "GDAL WMS driver (http://www.gdal.org/frmt_wms.html)" par défaut). Ajouté à GDAL 1.8.0
<Referer>http://example.foo/</Referer>	Chaîne de *Referer* HTTP. Certains serveurs peuvent le nécessiter (optionnel). Ajouter à GDAL 1.9.0
</GDAL_WMS>	

A Minipilote

Le pilote WMS de GDAL gère plusieurs 'minipilote' interne, qui permettent des accès à différents services de cartographiques web. Chacun de ces services peuvent gérer un ensemble différent d'options dans le bloc *Service*.

1 WMS

Communications avec un serveur WMS OGS. Possède la gestion pour les requêtes tuilées et non tuilées.

2 TileService

Service pour gérer la communication avec un service WorldWind. L'accès est toujours basé sur les tuiles.

3 WorldWind

Accès aux services web tuilé de WorldWind. L'accès est toujours basé sur les tuiles.

4 TMS (GDAL 1.7.0 et sup.)

Le mini-pilote TMS est structuré d'abord pour gérer l'utilisation des spécifications TMS. Ce service gère seulement l'accès aux tuiles.

Parce que le TMS est similaire à beaucoup d'autres faveurs de services 'x/y/z' sur le web, ce service peut également être utilisé pour accéder à ces services. Pour l'utiliser de cette façon, vous pouvez utiliser des variables de remplacement, de la forme ${x}, ${y}, etc.

Les variables gérées (le nom est sensible à la casse) sont :

- ${x} -- position x de la tuile

- ${y} -- position y de la tuile. Cela peut être soit le haut ou le bas de l' ensemble des tuile, basé sur le fait que le paramètre *YOrigin* est basé sur true ou false.
- ${z} -- position z de la tuile -- niveau de zoom
- ${version} -- paramètre de version, définie dans le fichier de config. 1.0.0 par défaut.
- ${format} -- format parameter, set in the config file. Defaults to 'jpg'.
- ${layer} -- layer parameter, set in the config file. Defaults to nothing.

Un *ServerURL* typique ressemblerait à cela :

```
http://labs.metacarta.com/wms-c/Basic.py/${version}/${layer}/${z}/${x}/${y}.${format}
```

Dans le but de mieux convenir aux utilisateurs du TMS, n'importe quel URL qui ne contient pas "${" aura automatiquement la chaîne ci-dessus (après "Basic.py/") rajoutée à leur URL.

Le service TMS a trois éléments de configuration XML qui sont différent des autres services : `Format` qui est définie par défaut à `jpg`, `Layer` qui n'as pas de valeur par défaut, et `Version` dont la valeur par défaut est `1.0.0`.

De plus, le service TMS respecte un paramètre supplémentaire, au niveau du *DataWindow*,qui est l'élément *YOrigin*. Cet élément doit être l'un parmi `bottom` (par défaut dans le TMS) ou `top`, qui correspond aux services OpenStreetMap et plus autres très populaires.

Deux exemples d'usage de service TMS sont présentés plus bas.

5 OnEarth Tiled WMS (GDAL 1.9.0 et sup.)

Le mini-pilote WMS tuilé d'OnEarth gère les spécifications WMS tuilés implémentées pour le pilote JPL d'OnEarth suivant la spécification http://onearth.jpl.nasa.gov/tiled.html.

Un fichier typique de configuration WMS tuilé d'OnEarth devrait ressembler à cela :

```
<GDAL_WMS>
    <Service name="TiledWMS">
    <ServerUrl>http://onmoon.jpl.nasa.gov/wms.cgi?</ServerUrl>
    <TiledGroupName>Clementine</TiledGroupName>
    </Service>
</GDAL_WMS>
```

La plupart des autres informations sont automatiquement récupérées du serveur distant en utilisant la méthode *GetTileService* au moment de l'ouverture.

6 VirtualEarth (GDAL 1.9.0 et sup.)

Accès au service de tuile par le web de Virtual Earth. L'accès est toujours basé sur les tuiles.

La variable *${quadkey}* doit être trouvé dans l'élément *ServerUrl*.

L'élément *DataWindow* peut être omis. Les valeurs par défaut sont :

- UpperLeftX = -20037508.34
- UpperLeftY = 20037508.34
- LowerRightX = 20037508.34
- LowerRightY = -20037508.34

- TileLevel = 19
- OverviewCount = 18
- SRS = EPSG:900913
- BlockSizeX = 256
- BlockSizeY = 256

B Exemples

- onearth_global_mosaic.xml - mosaic Landsat à partir du serveur WMS OnEarth

```
gdal_translate -of JPEG -outsize 500 250
onearth_global_mosaic.xml onearth_global_mosaic.jpg

gdal_translate -of JPEG -projwin -10 55 30 35 -outsize 500 250
onearth_global_mosaic.xml onearth_global_mosaic2.jpg
```

- metacarta_wmsc.xml - il est possible de configurer un service WMS se conformant à un cache WMS-C en définissant un nombre d'aperçue et la "taille du bloc" comme la taille de la tuile du cache. L'exemple suivant est un échantillon définie pour une cache WMS-C avec un "profile Global" de 19 niveau :

```
gdal_translate -of PNG -outsize 500 250 metacarta_wmsc.xml
metacarta_wmsc.png
```

- tileservice_bmng.xml - TileService, Blue Marble NG (Janvier)

```
gdal_translate -of JPEG -outsize 500 250 tileservice_bmng.xml
tileservice_bmng.jpg
```

- tileservice_nysdop2004.xml - TileService, NYSDOP 2004

```
gdal_translate -of JPEG -projwin -73.687030 41.262680 -73.686359
41.262345 -outsize 500 250 *
tileservice_nysdop2004.xml tileservice_nysdop2004.jpg
```

- Exemple du service TMS d'OpenStreetMap : se connecte au service de tuile d'OpenStreetMap. Notez que ce fichier permet l'utilisation du cache de tuile : plus d'information sur la configuration des paramètres du cache de tuile est disponible ci-dessus.

```
gdal_translate -of PNG -outsize 512 512
frmt_wms_openstreetmap_tms.xml openstreetmap.png</tt>
```

- Exemple de la couche TMS de MetaCarta, accède à la couche par défaut du TMS de TMS.

```
gdal_translate -of PNG -outsize 512 256
frmt_wms_metacarta_tms.xml metacarta.png</tt>
```

- Exemple BlueMarble sur Amazon S3 accédé avec le minipilote TMS.
- Google Maps accédé avec le minipilote TMS.
- Serveur carto de tuiles d'ArcGIS accédé avec le minipilote TMS.
- Cartes du géoportail Suisse accédé avec le minipilote TMS (nécessite GDAL >= 1.9.0)
- Exemples du WMS tuilé de OnEarth Clementine, journalier, et srtm.

- Couche Aerial de VirtualEarth accédé avec le minipilote VirtualEarth.

C Syntaxe ouverte

Le pilote WMS peut ouvrir :

- un fichier XML de description de service local : `gdalinfo description_file.xml`
- le contenu d'un fichier XML de description fournie comme nom de fichier :

```
gdalinfo "<GDAL_WMS><Service
name=\"TiledWMS\"><ServerUrl>http://onearth.jpl.nasa.gov/wms.cgi
? \
    </ServerUrl><TiledGroupName>Global SRTM
Elevation</TiledGroupName></Service></GDAL_WMS>"
```

- (GDAL >= 1.9.0) une URL d'un service WMS, préfixé avec *WMS:* :

```
gdalinfo "WMS:http://wms.geobase.ca/wms-bin/cubeserv.cgi"
```

  Une liste de sous jeu de données sera retournée, résultant de la lecture de la requête GetCapabilities sur ce serveur.
- (GDAL >= 1.9.0) une pseudo requête GetMap, telle que le nom du sous jeu de données l'a retournée par la syntaxe précédente :

```
gdalinfo "WMS:http://wms.geobase.ca/wms-bin/cubeserv.cgi?
SERVICE=WMS&amp;VERSION=1.1.1&amp; \
REQUEST=GetMap&amp;LAYERS=DNEC_250K:ELEVATION/ELEVATION&amp;SRS=
EPSG:42304&amp;BBOX=-3000000,-1500000,6000000,4500000"
```

- (GDAL >= 1.9.0) l'URL de base d'un service WMS tuilé, préfixé avec *WMS:* et avec l'argument GET *request=GetTileService* :

```
gdalinfo "WMS:http://onearth.jpl.nasa.gov/wms.cgi?
request=GetTileService"
```

  Une liste de sous jeu de données sera renvoyée, résultant de la lecture de la requête du GetTileService sur ce serveur.
- (GDAL >= 1.9.0) l'URL d'un service REST pour un serveur carto ArcGIS :

```
gdalinfo
"http://server.arcgisonline.com/ArcGIS/rest/services/World_Image
ry/MapServer?f=json&amp;pretty=true"
```

D Voir également

- OGC WMS Standards : http://www.opengeospatial.org/standards/wms
- Recommandation du WMS Tiling Client (WMS-C) : http://wiki.osgeo.org/index.php/WMS_Tiling_Client_Recommendation
- TileService WorldWind : http://www.worldwindcentral.com/wiki/TileService
- Spécification TMS
- Spécification WMS Tuilé OnEarth

Chapitre LXXVII XYZ -- ASCII Gridded XYZ

(Disponible à partir de GDAL >= 1.8.0)

GDAL gère la lecture et l'écritue de jeux de données raster XYZ **quadrillées** ASCII (ie. les XYZ non quadrillé, XYZ LIDAR, etc. doivent être ouvert par d'autres moyens. Voir la documentation de la commande :ref:`gdal.gdal.gdal_grid`).

Unknown interpreted text role "ref".

Ces jeux de données sont des fichiers ASCII avec (au moins) 3 colonnes, chaque lignes contenant les coordonnées X et Y du centre de la cellule et sa valeur.

L'espace entre chaque cellule doit être constante et aucune valeur manquante n'est gérée. Les cellules avec les mêmes coordonnées doivent être placées sur des lignes consécutives. Pour une même valeur de coordonnées Y, les lignes dans le jeu de données doit être organisé en augmentant les valeurs de X. La valeur des coordonnées Y peut augmenter ou diminuer cependant. Les séparateurs de colonnes gérés sont l'espace, les virgules, le point-virgule et les tabulations.

Le pilote tente de détecter une ligne d'en-tête et cherchera les noms 'x', 'lon' ou 'east' pour détecter l'index de la colonne X, 'y', 'lat' ou 'north' pour la colonne Y et 'z', 'alt' ou 'height' pour la colonne Z. Si aucun en-tête n'est présent ou une des colonnes ne peuvent être identifié dans l'en-tête, les colonnes X, Y et Z (dans cet ordre) sont supposé d'être les trois premières colonnes de chaque ligne.

L'ouverture d'un gros jeu de données peut être lente puisque le pilote doit scanner l'ensemble du fichier pour déterminer la taille du jeu de donnes et la résolution spatiale. Le pilote autodétectera le type de données parmi Byte, Int16, Int32 ou Float32.

A Options de création

- **COLUMN_SEPARATOR=a_value :** où a_value est une chaîne utilisée pour séparer la valeur des colonnes X,Y et Z. Par défaut à un espace
- **ADD_HEADER_LINE=YES/NO :** si une ligne d'en-tête doit être écrite (le contenu est X <col_sep> Y <col_sep> Z). NO par défaut

B Voir également

- Documentation de la commande :ref:`gdal.gdal.gdal_grid`

Unknown interpreted text role "ref".

Chapitre LXXVIII Les commandes OGR

Chapitre LXXIX Présentation d'OGR

La bibliothèque OGR Simple Features est une bibliothèque (et outils en ligne de commande) open source en C++ fournissant un accès en lecture (et parfois en écriture) d'une grande variété de formats de fichiers vecteur incluant les fichiers ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, et les formats Mapinfo mid/mif et TAB.

OGR fait partie de la bibliothèque :ref:`gdal.gdal.presentation`.

Unknown interpreted text role "ref".

A Ressource

- :ref:`gdal.ogr.formats.index` : ESRI Shapefile, ESRI ArcSDE, MapInfo (tab et mid/mif), GML, KML, PostGIS, Oracle Spatial, ...
  Unknown interpreted text role "ref".
- Programme OGR : ogrinfo, ogr2ogr, ogrtindex
- :ref:`gdal.ogr.sql`
  Unknown interpreted text role "ref".

B Download

1 Exécutable prêt à être utilisé

Duplicate explicit target name: "openev".

Le meilleur moyen pour obtenir les commandes OGR prêtes à l'emploi est de télécharger le dernier kit FWTools pour votre plateforme. Bien qu'importante, ceci inclus les compilations des commande OGR avec plusieurs composants optionnels. Une fois téléchargé suivez les instructions incluses pour configurer vos chemins et autres variables d'environnement correctement puis vous pourrez utiliser les différentes commandes OGR à partir de la ligne de commande. Le kit inclus également OpenEv, un visualiseur qui affichera les fichiers vecteurs gérés par OGR.

2 Source

Le code source de ce projet est disponible en OpenSource sous licence type X

Consortium. La bibliothèque OGR est pour le moment un sous-composant faiblement couplé de la bibliothèque GDAL. Vous obtenez tout GDAL pour le prix d'OGR. Voyez la page de téléchargement de GDAL et de compilation pour les détails pour obtenir les source et les compiler.

C Rapport de Bug

Les bugs de GDAL/OGR peuvent être remontés et listé en utilisant TRAC.

D Listes de diffusion

Une liste de faible volume gdal-announce où vous pouvez vous inscrire vous permet de recevoir des informations sur le projet GDAL/OGR.

La liste gdal-dev@lists.osgeo.org peut être utilisé pour les discussions sur le développement et les problèmes d'utilisation lié à OGR et aux technologies associées. L'inscription peut être réalisé et les archives visualiser sur le web.

Chapitre LXXX ogrinfo

Listes des informations sur une source de données géré par OGR.

Utilisation :

```
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 ...]]
```

Le programme `ogrinfo` liste diverses informations sur la source de données gérée par OGR vers la sortie standard (le terminal).

- **-ro :** ouvre la source de données en lecture seule.
- **-al :** liste tous les objets de toutes les couches (utilisé à la place de fournir des noms de couche comme arguments).
- **-so`` / ``Summary Only :** supprime la liste des objets, affiche seulement les informations comme la projection, le schéma, le nombre d'objets et l'étendue.
- **-q :** Rapport d'informations diverses en mode silence, incluant le système de coordonnées, le schéma de la couche, l'étendue et le nombre d'objets.
- **-where restricted_where :** un requête attributaire dans une forme restreinte sous la forme de requête SQL WHERE. Seulement les objets correspondant à la requête attributaire seront renvoyés.
- **-sql statement :** exécute la requête SQL indiquée et retourne le résultat.
- **-spat xmin ymin xmax ymax :** la zone d'intérêt. Seulement les objets à l'intérieur du rectangle seront renvoyés.
- **-fid fid :** si fournit, seulement les objets avec cet identifiant seront renvoyés. Fonctionnement exclusif aux requêtes attributaire ou SQL.

Note!

si vous voulez sélectionner plusieurs features basées sur leur feature_id, vous pouvez également utilisé le fait que le 'fid' est un champ spécial reconnu par . le SQL d'OGR. Donc, '-where "fid in (1,3,5)"' sélectionnera les features 1, 3 et 5.

- **-fields={YES/NO} :** (à partir de GDAL 1.6.0) Si définie à *NO*, le dump des feature n'affichera pas les valeurs des champs. La valeur par défaut est *YES*.
- **-geom={YES/NO/SUMMARY} :** (à partir de GDAL 1.6.0) Si définie à *NO*, le dump des feature n'affichera pas les géométries. Si définie à *SUMMARY*, seul le résumé des géométries sera affiché. Si définie à *YES*, la géométrie sera rapporté au format WKT complet de l'OGC. La valeur par défaut est *YES*.
- **--formats :** liste les pilotes des formats qui sont activés.
- **datasource_name :** La source de données à ouvrir. Peut être un nom de fichier, un répertoire ou un autre nom virtuel. Regardez la liste des formats vecteurs OGR pour les sources de données gérées.
- **layer :** une ou plusieurs noms de couches peuvent être reportés.

Si aucun nom de couches n'est passé, alors `ogrinfo` renverra une liste de couches disponibles (et leur type géométrique). Si le(s) nom(s) des couches est passé alors leurs étendues, système de coordonnées, le nombre d'objet ; le type géométrique, le schéma et toutes les géométries correspondant aux requêtes des paramètres seront renvoyés vers le terminal. Si aucun paramètre de requête n'est fourni, toutes les géométries seront renvoyées.

Les géométries sont renvoyées au format WKT de l'OGC.

Exemple retournant toutes les couches d'un fichier NTF :

```
% 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)
```

Exemple en utilisant une requête pour restreindre la sortie des objets d'une couche :

```
% 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)
```

Chapitre LXXXI ogr2ogr

Convertie des données *simple features* entre divers formats de fichiers.

Utilisation :

```
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 ...]]
```

Options avancées :

```
[-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]
[-segmentize max_dist] [-fieldTypeToString All|(type1[,type2]*)]
[-splitlistfields] [-maxsubfields val]
[-explodecollections] [-zfield field_name]
```

Ce programme peut être utilisé pour convertir des données *simple features* dans des formats de fichiers tout en réalisant des opérations diverses pendant le processus comme des sélections spatiales ou attributaires, la réduction d'ensemble d'attributs, la définition du système de coordonnées en sortie ou même la reprojection des objets pendant la translation.

- **-f format_name :** retourne le fichier au format, (ESRI Shapefile par défaut), des valeurs possibles sont :
  - **-f "ESRI Shapefile"**

    - **-f "TIGER"**
    - **-f "MapInfo File"**
    - **-f "GML"**
    - **-f "PostgreSQL"**
- **-append :** ajoute à la couche existante au lieu d'en créer une nouvelle.
- **-overwrite :** efface la couche en sortie et en recréer une vide.
- **-update :** ouvre une source de donnes existantes en mode mise à jour plutôt que d'essayer d'en créer une autre.
- **-select field_list :** liste séparé par une virgule de champs à partir de la couche en entrée à copier à la nouvelle couche. Un champ est ignoré s'il est déjà mentionné précédemment dans la liste même si la couche en entré possède des noms de champs dupliqués (toutes par défaut ; un champ est ignoré si un champ subséquent si un champ de même nom est trouvé).
- **-progress :** (à partie de GDAL 1.7.0) affiche un barre de progression dans la console. Fonctionne seulement si les couches en entrée possèdent la capacité "*fast feature count*".
- **-sql sql_statement :** requête SQL à exécuter. La couche/table résultante sera sauvé vers la sortie.
- **-dialect *dialect* :** dialecte SQL. Dans certains car permet d'utiliser le SQL d'OGR (non optimiser) au lieu du SQL natif d'un SGBDR en passant OGRSQL.
- **-where restricted_where :** requête attributaire (identique à la requête SQL WHERE)
- **-skipfailures :** continue après un échec, ignorant l'objet en échec.
- **-spat xmin ymin xmax ymax :** requête sur l'étendue spatiale. Seule les features dont les géométries intersectent les extends seront sélectionnés. Les géométries ne seront pas découpé sauf si l'option *-clipsrc* est spécifiée.
- **-dsco NAME=VALUE :** option de création du jeu de données (spécifique au format)
- **-lco NAME=VALUE :** option de création de couche (spécifique au format)
- **-nln name :** assigne un nom alternatif à la nouvelle couche.
- **-nlt type :** définie le type de géométrie pour la couche crée. Un parmi *NONE*, *GEOMETRY*, *POINT*, *LINESTRING*, *POLYGON*, *GEOMETRYCOLLECTION*, *MULTIPOINT*, *MULTILINE*, *MULTIPOLYGON* ou *MULTILINESTRING*. Ajouter "25D" au nom pour obtenir des versions en 2.5D.
- **-a_srs srs_def :** assigne un SRS en sortie.
- **-t_srs srs_def :** reprojète/transforme dans ce SRS en sortie.
- **-s_srs srs_def :** écrase la source SRS.
- **-fid fid :** si fournit, seulement l'objet avec cet identifiant sera renvoyé. Opère de façon exclusive aux requêtes spatiales ou attributaires.

Note!

si vous voulez sélectionner plusieurs features basées sur leur feature_id, vous pouvez également utilisé le fait que le 'fid' est un champ spécial reconnu par le SQL d'OGR. Donc, '-where “fid in (1,3,5)”' sélectionnera les features 1, 3 et 5.

Srs_def peut être une définition WKT complète (difficile d'échappé proprement), ou une définition *well known* (par exemple *EPSG:4326*) ou un fichier avec une définition WKT.

Options avancées :

- **-gt n :** regroupe n objets par transaction (200 par défaut).
- **-clipsrc [xmin ymin xmax ymax]|WKT|datasource|spat_extent :** (à partir de

GDAL 1.7.0) (à partir de GDAL 1.7.0) découpe les géométries dans la bounding box définie (exprimée dans la projection source), géométrie WKT (POLYGON ou MULTIPOLYGON), à partir d'une source de données ou de l'étendue sptiale de l'option *-spat* si vous utilisez le mot clé *spat_extent*. Lors de l'utilisation d'une source de données, vous désirez généralement l'utiliser en combinaison des options -clipsrclayer, -clipsrcwhere ou -clipsrcsql

- **-clipsrcsql sql_statement :** sélectionne les géométries désirées en utilisant une requête SQL à la place.
- **-clipsrclayer layername :** sélectionne le nom de la couche à partir de la source de données source du clip.
- **-clipsrcwhere expression :** restreint les géométries désirées basées sur une requête attributaire.
- **-clipdst xmin ymin xmax ymax :** (à partir de GDAL 1.7.0) découpe les géométries après la reprojection avec la bounding box définie (exprimé en SRS destinataire), géométrie WKT (POLYGON ou MULTIPOLYGON) ou à partir d'une source de données. Lorsqu'une source de données est définie, vous voudrez généralement l'utiliser en combinaison des options *-clipdstlayer*, *-clipdstwhere* ou *-clipdstsql*.
- **-clipdstsql sql_statement :** sélectionne les géométries désirées en utilisant une requête SQL à la place.
- **-clipdstlayer layername :** sélectionne le nom de la couche à partir de la source de données de destination du clip.
- **-clipdstwhere expression :** restreint les géométries désirées basées sur une requête attributaire.
- **-wrapdateline :** (à partir de GDAL 1.7.0) découpe les géométries qui croise le méridien "final" (long. = +/- 180deg)
- **-segmentize max_dist :** (à partir de GDAL 1.6.0) distance maximale entre deux noeuds. Utilisé pour créer des points intermédiaires.
- **-fieldTypeToString type1, ... :** (à partir de GDAL 1.7.0) convertie un champs du type définie vers le type String dans la couche de destination. Les types valides sont : Integer, Real, String, Date, Time, DateTime, Binary, IntegerList, RealList, StringList. La valeur spéciale *All* peut être utilisée pour convertir tous les champs en String. C'est une manière alternative pour utiliser l'opérateur *CAST* du SQL d'OGR, qui peut éviter d'entrer une longue requête SQL.
- **-splitlistfields :** (à partir de GDAL 1.8.0) découpe les champs de type StringList, RealList ou IntegerList dans autant de champs de type String, Real ou Integer que nécessaire.
- **-maxsubfields val :** pour combiné avec l'option *-splitlistfields* pour limiter le nombre de sous champs créé pour chaque champs découpés.
- **-explodecollections :** (à partir de GDAL 1.8.0) produit un feature pour chaque géométrie dans n'importe quelle collection géométrique du fichier source.
- **-zfield *field_name* :** (à partir de GDAL 1.8.0) utilise le champ définie pour remplir les coordonnées Z des géométries.

A Exemples

Exemple ajoutant une couche existante (les deux options nécessites d'être utilisé) :

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

Exemple reprojetant les données à partir de ETRS_1989_LAEA_52N_10E vers EPSG:4326 et découpant les features par une bounding box :

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

Des exemples supplémentaires sont données dans les pages des formats.

Chapitre LXXXII ogrtindex

Créer un index de tuile.

Utilisation :

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

Duplicate explicit target name: "mapserver".

Le programme `ogrtindex` peut être utilisé pour créer des index de tuiles - un fichier contenant une liste d'identification d'un groupe d'autre fichiers en fonction de leurs étendues spatiales. Cela a d'abord pour but d'être utilisé avec MapServer pour l'accès aux tuiles en utilisant le type de connexion ogr.

- **-lnum n :** ajoute le numéro de couche 'n' de chaque fichier source dans l'index des tuiles.
- **-lname nom :** ajoute la couche nommé 'nom' de chaque fichier source dans l'index des tuiles.
- **-f output_format :** sélectionne un nom de format en sortie. Créer un shapefile par défaut.
- **-tileindex field_name :** le nom à utiliser pour le nom du jeu de données. *LOCATION* par défaut.
- **-write_absolute_path :** les noms de fichiers sont écrit avec des chemins absolus.
- **-skip_different_projection :** seulement les couches avec le même référentiel de projection que des couches déjà inséré dans l'index des tuiles seront inséré.

Si aucun arguments *-lnum* ou *-lname* sont données, il est supposé que toutes les couches dans les jeux de données sources doivent être ajoutées à l'index de tuile comme enregistrement indépendant.

Si l'index de tuile existe déjà, les enregistrements seront ajouté, autrement l'index sera crée.

Duplicate explicit target name: "mapserver".

C'est un des défauts de l'actuel programme `ogrtindex` qui ne cherche pas à copier la définition du système de coordonnées de la source de données vers l'index de tuiles (comme il est attendue par MapServer quand *PROJECTION AUTO* est utilisé).

Cet exemple créera un shapefile (tindex.shp) contenant un index de tuiles des couches *BL2000_LINK* dans tous les fichiers NTF dans le répertoire de travail :

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

Chapitre LXXXIII SQL dans OGR

OGRDataSource gère l'exécution de commandes en opposition à la source de données via la méthode *OGRDataSource::ExecuteSQL()*. Bien qu'en théorie n'importe quelle commande peut être prise en charge de cette manière, en pratique le mécanisme est utilisé pour fournir un sous ensemble des possibilités de SELECT de SQL aux applications. Cette page discute de l'implémentation de SQL générique dans OGR, et des problèmes avec la gestion des SQL spécifique au pilote.

A Syntaxe SQL gérée

La classe *OGRLayer* gère également l'application d'un filtre de requête attributaire aux features retournées en utilisant la méthode *OGRLayer::SetAttributeFilter()*. La syntaxe pour le filtre attributaire est la même que la clause WHERE dans la requête SELECT du SQL d'OGR. Donc tout ce qui concerne ici la clause WHERE s'applique également dans le contexte de la méthode *SetAttributeFilter()*

Note!

OGR SQL a été implémenté pour la version 1.8.0 de GDAL/OGR. Plusieurs fonctionnalités présentées ci-dessous, notamment les expressions arithmétiques et les expressions dans la liste des champs, n'étaient pas gérées dans la version 1.7.x ou plus ancienne de GDAL/OGR. Voyez la RFC 28 pour les détails des nouvelles fonctionnalités dans la version 1.8.0 de GDAL/OGR.

B SELECT

La requête SELECT est utilisée pour récupérer les objets d'une couche (analogue aux lignes des tables dans un RDBMS) avec le résultat de la requête représentée comme une couche temporaire d'objets. Les couches de la source de données sont analogues aux tables dans un RDBMS et les attributs des objets sont analogues aux valeurs des colonnes. La forme la plus simple d'une requête SELECT du SQL d'OGR ressemble à cela :

```
SELECT * FROM polylayer
```

Dans ce cas tous les objets sont récupérés de la couche nommée *polylayer*, et tous les attributs de ces objets sont renvoyés. C'est essentiellement équivalent à accéder à la couche directement. Dans cet exemple l'"*" est la liste de tous les champs à récupérer de la couche, avec "*" signifiant que tous les champs sont récupérés.

Cette forme sensiblement plus sophistiquée renvoie encore tous les objets d'une couche mais le schéma contiendra les attributs *EAS_ID* et *PROP_VALUE*. N'importe quel attribut sera ignoré.

```
SELECT eas_id, prop_value FROM polylayer
```

Un SELECT un petit peu plus ambitieux, restreignant les objets récupérés avec une clause WHERE et classant les résultats, ressemblera à ceci :

```
SELECT * from polylayer WHERE prop_value > 220000.0 ORDER BY
prop_value DESC
```

Cette requête SELECT produira une table avec juste un objet, avec un attribut (nommé *count_eas_id*) contenant le nombre de valeur distincte des attributs *eas_id*.

```
SELECT COUNT(DISTINCT eas_id) FROM polylayer
```

1 Opérateurs de liste de champs

La liste de champs est une liste séparée par des virgules de champs pour rapporter les objets en sortie de la couche source. Ils apparaitront sur les objets en sortie dans l'ordre où ils apparaissent dans la liste des champs, cette liste peut donc être utilisé pour ré-ordonner les champs.

Une forme spéciale de la liste des champs utilise le mot-clé DISTINCT. Cela renvoie une liste de valeurs distinctes de l'attribut nommé. Quand le mot-clé DISTINCT est utilisé, seulement un attribut peut apparaitre dans la liste. Ce mot-clé peut être utilisé avec n'importe quel type de champ. Pour l'instant le test pour faire la distinction entre les valeurs est sensible à la casse dans le SQL d'OGR. Le résultat d'un SELECT avec le mot-clé DISTINCT est une couche avec une colonne (nommé de la même manière que le champs sur lequel la sélection s'opère), et un objet par valeur distinct. Les géométries sont ignorées. Les valeurs distinctes sont assemblée en mémoire, donc cela peut utiliser beaucoup de mémoire pour des jeux de données avec un grand nombre de valeurs distinctes.

```
SELECT DISTINCT areacode FROM polylayer
```

Il y a également plusieurs opérateurs de synthèse qui peuvent être appliqués aux colonnes. Quand un opérateur de synthèse est appliqué à un champ, alors un opérateur de synthèse doit être appliqué sur tous les champs. Les opérateurs de synthèses sont COUNT (compte le nombre d'instance), AVG (moyenne arithmétique), SUM (somme numérique), MIN (sémantique ou minimum numérique) , et MAX (sémantique ou maximum numérique). Cet exemple produit diverses informations de synthèse sur les valeurs des propriétés des parcelles :

```
SELECT MIN(prop_value), MAX(prop_value), AVG(prop_value),
```

```
SUM(prop_value),
   COUNT(prop_value) FROM polylayer WHERE prov_name = "Ontario"
```

Un cas spécial, on peut donner l'argument "*" à l'opérateur COUNT() à la place du nom du champs qui est une forme raccourcit pour compter tous les enregistrements bien qu'il donnera le même résultat en utilisant n'importe quels noms de colonne. Il est également possible d'appliquer l'opérateur COUNT() à un SELECT DISTINCT pour obtenir le nombre de valeurs distinctes, par exemple :

```
SELECT COUNT(DISTINCT areacode) FROM polylayer
```

Les noms des champs peuvent également être préfixé par le nom d'une table bien que cela soit réellement significatif que pour les jointures. Cela est démontré plus loin dans la section JOIN.

Les définitions de champs peuvent aussi être des expressions complexes en utilisant des opérateurs arithmétiques et fonctionnels. Cependant, le mot-clé DISTINCT, et les opérateurs d'agrégation MIN, MAX, AVG et SUM ne peuvent pas être appliqués aux expressions de champs.

```
SELECT cost+tax from invoice
```

Literal block ends without a blank line; unexpected unindent.

ou

```
SELECT CONCAT(owner_first_name,' ',owner_last_name) from properties
```

2 Utiliser les alias des noms de champs

SQL d'OGR gère le renommage des champs en suivant la spécifications SQL92 en utilisant le mot-clé AS comme pour l'exemple suivant :

```
SELECT *, OGR_STYLE AS 'STYLE' FROM polylayer
```

L'alias du nom du champ peut être utilisé comme la dernière opération dans la spécification de la colonne. Par conséquent nous ne pouvons pas renommer les champs à l'intérieure d'un opérateur, mais nous pouvons renommer toute l'expression de la colonne, comme ces deux exemples :

```
SELECT COUNT(areacode) AS 'count' FROM polylayer
SELECT dollars/100.0 AS cents FROM polylayer
```

3 Changer le type des champs

À partir de GDAL 1.6.0, SQL d'OGR gère le changement du type des colonnes en utilisant l'opérateur CAST conforme SQL92 comme pour l'exemple suivant :

```
SELECT *, CAST(OGR_STYLE AS character(255)) FROM rivers
```

Pour l'instant la transformation vers les cibles suivantes sont gérées :

- *character(field_length)*, field_length=1 par défaut
- *float(field_length)*
- *numeric(field_length, field_precision)*
- *integer(field_length)*
- *date(field_length)*
- *time(field_length)*
- *timestamp(field_length)*

Définir *field_length* et/ou *field_precision* est optionel. Une valeur zéro explicite peut être utilisée comme la largeur d'un champ character() pour indiquer la largeur de la variable. La conversion vers les types de données OGR 'liste d'entier', 'liste double' et 'liste de caractères' ne sont pas gérés, ce qui n'est pas conforme aux spécification SQL92.

Bien que l'opérateur CAST peut être appliqué n'importe où dans une expression, dont la clause WHERE, le contrôle du format du champ en sortie est seulement géré si l'opérateur CAST est l'opérateur le plus à l'extérieur sur un champ dans une liste de définition de champs. Dans d'autres contexte il est encore utile de convertir entre les types de donnée numérique, chaîne et date.

4 WHERE

L'argument de la clause WHERE est une expression logique assez simpliste utilisé pour sélectionner les enregistrements d'une couche source. En plus de cette utilisation dans la requête WHERE, la prise en charge de la clause WHERE est également utilisé par les requêtes attributaires d'OGR sur les couches normales via *OGRLayer::SetAttributeFilter()*.

En plus des opérateurs arithmétiques et autres opérateurs fonctionnels disponibles dans l'expression dans la clause de définition des champs de la requête SELECT, les opérateurs logiques sont aussi disponible dans la clause WHERE et la valeur évaluée de l'expression doit être logique(true ou false).

Les opérateurs logiques disponibles sont =, !=, <>, <, >, <=, >=, LIKE, ILIKE, BETWEEN et IN.

La plupart des opérateurs s'expliquent par eux-mêmes, mais il n'est pas évident que ''!='' ne soit pas équivalent à ''<>'', la chaine égalité n'est pas sensible à la casse, mais les opérateurs <, >, <= et >= sont sensible à la casse. À la fois LIKE et ILIKE sont insensible à la casse.

L'argument valeur à l'opérateur LIKE est un motif avec lequel la chaine de valeur est recherché. Dans ce motif le signe pourcentage (%) correspond à un nombre de caractères, et underscore (_) correspond à un seul caractère. Une clause optionnelle ESCAPE *escape_char* peut être ajoutée afin que les caractères % ou _ puissent être recherchés comme caractères normaux, en étant précédé de *escape_char*.

String	***Pattern***	***Matches?***
Alberta	ALB%	Yes
Alberta	_lberta	Yes
St. Alberta	_lberta	No
St. Alberta	%lberta	Yes

Robarts St.	%Robarts%	Yes
12345	123%45	Yes
123.45	12?45	No
N0N 1P0	%N0N%	Yes
L4C 5E2	%N0N%	No

L'opérateur IN prendre une liste de valeur comme argument et teste la présence dans cet ensemble de la valeur de l'attribut.

Value	***Value Set***	***Matches?***
321	IN (456,123)	No
"Ontario"	IN ("Ontario","BC")	Yes
"Ont"	IN ("Ontario","BC")	No
1	IN (0,2,4,6)	No

La syntaxe de l'opérateur BETWEEN est "*field_name BETWEEN value1 AND value2*" et il est équivalent à "*field_name >= value1 AND field_name <= value2*".

En plus des opérateurs binaire ci-dessus, il y a des opérateurs additionnels pour tester si un champ est null ou pas. Ce sont les opérateursIS NULL et IS NOT NULL.

Les tests de champ basic peuvent être combiné dans des prédicats plus compliqué en utilisant les opérateurs logique AND, OR, et le prédicat logique unaire NOT. Les sous-expressions doivent être mis entre parenthèse pour permettre une claire priorité. quelques prédicats plus compliqués :

```
SELECT * FROM poly WHERE (prop_value >= 100000) AND (prop_value <
200000)
SELECT * FROM poly WHERE NOT (area_code LIKE "N0N%")
SELECT * FROM poly WHERE (prop_value IS NOT NULL) AND (prop_value <
100000)
```

5 Limitations de la clause WHERE

- Les champs doivent tous venir de la table primaire (celle listée dans la clause FROM.
- Toutes les comparaisons de chaine sont insensible à la casse sauf pour <, >, <= et >=.

6 ORDER BY

La clause ORDER BY est utilisé pour forcer les objets renvoyés à être ordonné (ascendant ou descendant) sur un des champs. L'ordre ascendant (augmentant) est celui par défaut si aucun des mot-clés ASC ou DESC n'est fournie. Par exemple :

```
SELECT * FROM property WHERE class_code = 7 ORDER BY prop_value DESC
```

```
SELECT * FROM property ORDER BY prop_value
SELECT * FROM property ORDER BY prop_value ASC
SELECT DISTINCT zip_code FROM property ORDER BY zip_code
```

Notez que les clauses ORDER BY entraine de passage sur l'ensemble des objets. Le premier pour construire la table des valeurs correspondantes des champs en mémoire avec l'id des objets, et le second passage pour récupérer les objets par id dans l'ordre. Pour les formats dont les id des objets ne peuvent pas être lu efficacement d'une manière aléatoire cela peut être une opération couteuse.

L'ordonnancement de valeurs de champs de type chaine est sensible à la casse, et pas insensible à la casse comme dans la plupart des cas dans SQL d'OGR.

7 Clause JOIN

SQL d'OGR gère une forme limité de jointure une à une. Cela permet à des enregistrements d'une table secondaire d'être utilisé pour la recherche avec une clé partagée entre elle et la table primaire lors d'une requête. Par exemple, une table de location de ville pourrait inclure une colonne *nation_id* qui peut être utilisé comme référence dans une table *nation* secondaire pour récupérer les noms des pays. Une requête par jointure pourrait ressembler à ceci :

```
SELECT city.*, nation.name FROM city
 LEFT JOIN nation ON city.nation_id = nation.id
```

Cette requête renverrait une table avec tous les champs de la table *city*, et un champ supplémentaire *nation.name* avec le pays à l'intérieur récupérer de la table *nation* en cherchant les enregistrements dans la table nation qui ont le champ *id* avec la même valeur que le champ *city.nation_id*.

Les jointures introduisent des problèmes supplémentaires. Parmi ceux là le concept de référencement de table sur les noms de champ. Par exemple, se référer à *city.nation_id* plutôt que juste *nation_id* pour indiquer le champ *nation_id* de la couche *city*. La référence du nom de la table peut seulement être utilisé dans la liste des champs, et dans la clause ON d'une jointure.

Les caractères de substitution sont parfois impliqué. Tous les champs d'une table primaire (*city* dans notre cas) et la table secondaire (*nation* dans ce cas) peuvent être sélectionné en utilisant le caractère * de substitution. Mais les champs d'une seul table primaire ou secondaire peuvent être sélectionné en préfixant l'astérix avec le nom de la table.

Les noms des champs dans la couche de la requête résultante sera qualifié du nom de la table, si le nom de la table est donné comme référence dans la liste des champs. De plus les noms des champs seront qualifiés avec un nom de table s'ils ne rentrent pas en conflit avec un nom de champs existant. Par exemple, la requête select suivante pourrait résulter dans un ensemble de champ *name*, *nation_id*, *nation.nation_id* et *nation.name* si les tables *city* et *nation* ont tout deux le champs *nation_id* et *names*.

```
SELECT * FROM city LEFT JOIN nation ON city.nation_id =
nation.nation_id
```

D'un autre côté si la table *nation* a un champ *continent_id* mais pas la table *city*, alors ce

champs ne nécessitera pas d'être référencé dans l'ensemble de résultat. Cependant, si la requête select ressemble à la commande suivante, tous les champs résultant seront référencés par le nom de la table :

```
SELECT city.*, nation.* FROM city
    LEFT JOIN nation ON city.nation_id = nation.nation_id
```

Dans les exemples au-dessus, la table *nation* a été trouvé dans la même source de données que la table *city*. Cependant, la gestion de jointure d'OGR inclus la possibilité de joindre une table dans une source de données différente, éventuellement d'un format différent. Cela est indiqué en référençant la table secondaire avec le nom d'une source de données. Dans ce cas la source de données secondaire est ouverte en utilisant une sémantique normale d'OGR et utilisée pour accéder à la table secondaire jusqu'à ce que le résultat de la requête n'est plus nécessaire.

```
SELECT * FROM city
    LEFT JOIN '/usr2/data/nation.dbf'.nation ON city.nation_id =
nation.nation_id
```

Bien que pas forcément nécessaire, il est également possible d'introduire des alias de table pour simplifier certaines requêtes SELECT. Cela peut aussi être utile pour enlever tout ambigüité lorsque des tables de même noms sont utilisé de différents sources de données. Par exemple, si les noms des tables réels n'étaient pas soignées nous voudrions réaliser quelque chose comme :

```
SELECT c.name, n.name FROM project_615_city c
    LEFT JOIN '/usr2/data/project_615_nation.dbf'.project_615_nation n
        ON c.nation_id = n.nation_id
```

Il est possible de réaliser des jointures multiples dans une seule requête :

```
SELECT city.name, prov.name, nation.name FROM city
    LEFT JOIN province ON city.prov_id = province.id
    LEFT JOIN nation ON city.nation_id = nation.id
```

8 Limitations de la clause JOIN

- Les jointures peuvent être des opérations couteuses si la table secondaire n'est pas indexée sur le champ clé de la jointure.
- Les champs joins ne peuvent pas être utilisés dans les clauses WHERE, ou ORDER BY en même temps. La jointure est essentiellement évalué après que tous les sous-ensemble des tables primaires soient complète et après le passage du ORDER BY.
- Les champs joins ne peuvent pas être utilisé comme clé dans une future jointure. Vous ne pouvez donc pas utiliser l'id de la province dans une ville pour rechercher les données de la provinces, puis utiliser un id d'un pays à partir de la province pour récupérer les données du pays. Cela est quelque chose qui pourrait être développé, mais n'est pas actuellement géré.
- Les noms des sources de données pour les tables jointes sont évalué par rapport au répertoire de travail du processus en cours, et pas du chemin de la source de données primaire.

- Il n'y a pas de réelle jointure LEFT ou RIGHT au sens RDBMS. Qu'un enregistrement secondaire existe ou non, une et une seule copie de l'enregistrement primaire est renvoyée dans l'ensemble des résultats. Si un enregistrement secondaire nepeut pas être trouvé, les champs dérivés secondaires sera NULL. Si plus d'une correspondance du champs secondaire est trouvé, seul le premier enregistrement sera utilisé.

C Champs spéciaux

Le processeur de requête SLQ d'OGR traite certains attributs d'objets comme des champs spéciaux interne et peuvent être utilisé dans les requêtes SQL comme tout autres champs. Ces champs peuvent être placé dans la liste des select, les clauses WHERE et ORDER BY. Les champs spéciaux ne seront pas inclus dans le résultat par défaut mais ils peuvent être explicitement inclus en les ajoutant à la liste du select. Lors de l'accès à la valeur du champ les champs spéciaux prennent la priorité sur tous les autres champs avec le même nom dans la source de données.

1 FID

Normalement l'id de l'objet est une propriété spéciale d'un objet et n'est pas traité comme un attribut d'objet. Dans certains cas il est pratique de pouvoir utiliser l'id de l'objet dans des requêtes et des résultats comme un champ normal. Pour cela utiliser le nom FID. L'utilisation du caractère de substitution de champ n'inclura pas l'id de l'objet, mais il peut être explicitement inclus en utilisant la syntaxe suivante :

```
SELECT FID, * FROM nation
```

2 OGR_GEOMETRY

Certaines sources de donnés (comme les fichiers tab de MapInfo) peuvent prendre en charge des géométries de différents types dans la même couche. Le champ spécial *OGR_GEOMETRY* représente le type de géométrie renvoyé par la méthode *OGRGeometry::getGeometryName()* et peut être utilisé pour distinguer les différents types. En utilisant ce champ on peut sélectionner des types particulier des géométries :

```
SELECT * FROM nation WHERE OGR_GEOMETRY='POINT' OR
OGR_GEOMETRY='POLYGON'
```

3 OGR_GEOM_WKT

La représentation *Well Known Text* d'une géométrie peut aussi être utilisé comme champ spécial. Pour sélectionner le WKT d'une géométrie *OGR_GEOM_WKT* peut être inclus dans la liste de select :

```
SELECT OGR_GEOM_WKT, * FROM nation
```

En utilisant *OGR_GEOM_WKT* et l'opérateur LIKE dans la clause WHERE nous pouvons avoir des effets similaire à l'utilisation de *OGR_GEOMETRY* :

```
SELECT OGR_GEOM_WKT, * FROM nation WHERE OGR_GEOM_WKT
```

```
   LIKE 'POINT%' OR OGR_GEOM_WKT LIKE 'POLYGON%'
```

4 OGR_GEOM_AREA

(à partir de GDAL 1.7.0)

Le champ spécial **OGR_GEOM_AREA** retourne la surface de la géométrie de la feature calculée par la méthode *OGRSurface::get_Area()*. Pour *OGRGeometryCollection* et *OGRMultiPolygon* la valeur est la somme des surface de ses membres. Pour les géométries non surfacique la surface retournée est 0.0.

Par exemple, pour sélectionner seulement les polygones plus grand qu'une surface donnée :

```
SELECT * FROM nation WHERE OGR_GEOM_AREA > 10000000'
```

5 OGR_STYLE

Le champs spécial *OGR_STYLE* représente la chaine de style d'un objet renvoyé par la méthode *OGRFeature::GetStyleString()*. En utilisant ce champ et l'opérateur LIKE le résultat d'une requête peut être filtré par le style. Par exemple nous pouvons sélectionner l'objet annotation avec :

```
SELECT * FROM nation WHERE OGR_STYLE LIKE 'LABEL%'
```

D CREATE INDEX

Certains pilotes SQL d'OGR gère la création d'indexes attributaires. Pour l'instant cela inclus le pilote Shapefile. Un inde accélère les requêtes attributaires de la forme *nomChamp = valeur,* ce qui est utilisé par la jointure. Pour créer un index attributaire sur le champs *nation_id* de la table *nation* une commande telle que celle-ci peut être utilisée :

```
CREATE INDEX ON nation USING nation_id
```

1 Limitations des Index

- Les index ne sont pas maintenu dynamiquement lors de l'ajout ou la suppression d'une couche d'un nouvel objet.
- Les chaines très longue (plus longue que 256 caractères) ne peuvent pas pour l'instant être indexé.
- Pour recréer un index il est nécessaire de supprimer tous les indexes sur la couche et de toutes les recréer.
- Les indexes ne sont pas utilisés dans toutes les requêtes complexes. Pour l'instant la seule requête qui sera accélérée est la requête simple *champ = valeur*.

E DROP INDEX

La commande SQL d'OGR DROP INDEX peut être utilisé pour supprimer tous les indexes

sur une table particulière ou juste l'index d'une colonne particulière.

```
DROP INDEX ON nation USING nation_id
DROP INDEX ON nation
```

F ExecuteSQL()

SQL est exécuté en fonction de *OGRDataSource,*, et pas en fonction d'une couche spécifique. L'appel ressemblera à ceci :

```
OGRLayer * OGRDataSource::ExecuteSQL( const char *pszSQLCommand,
                                      OGRGeometry *poSpatialFilter,
                                      const char *pszDialect );
```

L'argument *pszDialect* a pour objectif théorique de permettre la gestion de différents langages de commande en fonction d'un provider, mais pour l'instant les applications doivent toujours passer une chaine vide (pas NULL) pour avoir le dialecte par défaut.

L'argument *poSpatialFilter* est une géométrie utilisé pour sélectionner un rectangle de limite pour les objets à renvoyés d'une manière similaire à la méthode *OGRLayer::SetSpatialFilter()*. Il peut être NULL pour aucune restriction spatiale.

Le résultat d'un appel *ExecuteSQL()* est habituellement un OGRLayer temporaire représentant l'ensemble des résultats de la requête. C'est le cas des requêtes SELECT par exemple. La couche temporaire renvoyée doit être publiée avec la méthode *OGRDataSource::ReleaseResultsSet()* quand elle n'est plus nécessaire. L'échec de la publication avant que la source de données ne soit détruite entrainera un crash.

G SQL hors OGR

Tous les pilotes d'OGR pour les systèmes de bases de données : MySQL, PostgreSQL et PostGIS (:ref:`gdal.ogr.formats.postgresql`), Oracle (:ref:`gdal.ogr.formats.oci`), SQLite, ODBC; les Géodatabases Personnelles d'ESRI (:ref:`gdal.ogr.formats.pgeo`) et MS SQL Spatial (:ref:`gdal.ogr.formats.mssqlspatial`) écrasent la fonction *OGRDataSource::ExecuteSQL()* par une implémentation dédiée et, par défaut, envoie les requêtes SQL directement au RDBMS sous-jacent. Dans ces cas la syntaxe SQL varie plus ou moins du SQL d'OGR. Aussi, tout ce qui est possible en SQL peut alors être accomplie pour ces bases de données particulières. Seul le résultat des requêtes WHERE SQL sera renvoyé comme des couches.

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

Chapitre LXXXIV Les formats OGR

Chapitre LXXXV Aeronav FAA

(GDAL/OGR >= 1.8.0)<p>

This driver reads text files describing aeronav information - obstacles, navaids and routes - as provided by the FAA.<p>

A Voir également

- Fichier Digital Obstacle
- Fichier données NAVAID Digital
- Supplément Digital Aeronautical Chart

Chapitre LXXXVI ArcObjects d'ESRI

A Aperçu

Le pilote ArcObjects d'OGR fournit un accès en lecture seule vers les sources de données ArcObjects. Puisqu'il utilise le SDK d'ESRI, il est nécessaire d'avoir une licence ESRI pour fonctionner. Néanmoins, cela signifie également que le pilote a une complète abstraction d'ESRI. Parmi ces derniers, vous avez :

- GeoDatabases :
  - GeoDatabase Personnelle (.mdb)
  - fichier GeoDatabase (.gdb)
  - GeoDatabase Entreprise (.sde).
- Shapefiles d'ESRI

Bien que cela n'ait pas été étendue pour faire cela encore (il n'y a pas eut de besoin), il peut potentiellement géré également les abstractions GeoDatabase suivantes.

- Les classes d'entités d'annotation et de dimension
- Classes de relations
- Réseaux (GN et ND)
- Topologies
- Terrains
- Représentations
- Parcel Fabrics

Vous pouvez essayer ceux-ci et ils devraient fonctionner - mais n'ont pas été testés. Notez que les abstractions au-dessus ne peuvent pas être gérés avec l'API FileGeoDatabase ouverte.

B Dépendances

- Une licence ArcView ou ArcEngine (ou supérieur) - est nécessaire pour que cela fonctionne.
- Les bibliothèques ESRI installées. Cela est typiquement le cas si vous avez installé ArcEngine, ArcGIS Desktop ou Server - Nécessaire pour compiler. Notez que ce code devrait également compiler en utilisant le SDK *nix ArcEngine, cependant le développeur n'a pas accès à ce SDK et n'a donc pas pu l'essayer.

C Usage

Préfixé la source de données avec "AO:"

- Lire un fichier GDB et charger les données dans PostGIS :

```
ogr2ogr -overwrite -skipfailures -f "PostgreSQL" PG:"host=myhost
user=myuser
dbname=mydb password=mypass" AO:"C:\somefolder\BigFileGDB.gdb"
"MyFeatureClass"
```

- Obtenir des informations détaillées d'une GéoDatabase Personnelle :

```
ogrinfo -al AO:"C:\somefolder\PersonalGDB.mdb"
```

- Obtenir des informations détaillées de la GéoDatabase Enterprise (.sde contient une version cible auquel se connecter) :

```
ogrinfo -al AO:"C:\somefolder\MySDEConnection.sde"
```

D Notes de compilation

Lisez :ref:`gdal.install`. Vous trouverez une section similaire dans *nmake.opt* pour ArcObjects.

Unknown interpreted text role "ref".

Après cela, allez dans le répertoire *$gdal_source_rootogrogrsf_frmtsarcobjects* et exécutez :

```
nmake /f makefile.vc plugin
nmake /f makefile.vc plugin-install
```

E Problèmes connus

Les champs *date* et *blob* n'ont pas été implémentés. C'est probablement juste quelques lignes de code, mais le développeur n'a pas eut assez de temps.

Chapitre LXXXVII Arc/Info Binary Coverage

Les couvertures Arc/Info Binary (par exemple Arc/Info V7 et plus récente) sont gérées par OGR en lecture.

Les sections *label*, *arc*, *polygone*, *centroide*, *région* et *texte* d'une couverture sont toutes gérées comme couches. Les attributs provenant d'INFO sont ajoutés aux *labels*, *arc*, *polygones* ou *région* aux endroits appropriés. Lorsqu'elles sont disponibles, les informations de projections sont lu et traduit. Les géométries polygonales sont collectées pour les couches polygones et région à partir des arcs les composants.

Les sections textes sont représentées comme des couches ponctuelles. La hauteur d'affichage est préservée dans le champ attributaire *HEIGHT* ; cependant, d'autre information sur l'orientation du texte est ignorée.

Les tables attributaires associées avec une couverture, mais pas nommément désignés, qui doit être attachée à l'une des couches géométriques n'est pas accessible par OGR. Notez que les tables attributaires sont stockées dans un répertoire *info/* au même niveau que le répertoire de couverture. Si cela est inaccessible ou corrompue aucune information d'attributs ne sera ajoutée aux couches de couverture, mais la géométrie doit être encore accessible.

Si le répertoire contient des fichiers avec des noms comme w001001.adf alors la couverture est une :ref:`gdal.gdal.divers_formats.aig` qui peut être lu par GDAL, et n'est pas une couche vecteur gérée par OGR.

Unknown interpreted text role "ref".

Les couches sont nommées comme suit :

- une couche label (label de polygone, ou des points libres) est nommée *LAB* si présente.
- une couche centroide (centroide de polygone) est nommée *CNT* si présente.
- une couche arc (ligne) est nommée *ARC* si présente.
- une couche polygone est nommée *PAL* si présente.
- une section texte est nommé selon la sous-classe de la section.
- une sous-classe de région est nommé selon le nom de la sous-classe.

Le pilote des couvertures binaire d'Arc/Info tente d'optimiser les requêtes spatiales mais dû à l'absence d'index spatial cela est juste réalisé en minimisant les traitements pour les objets en dehors de la fenêtre spatiale.

La lecture aléatoire (par FID) des arcs, et des polygones est gérée, il peut ne pas être géré pour les autres types d'objets.

A Voir également

- Page de la bibliothèque AVCE00
- Pilote d'OGR pour AVCE00 (.E00)

Chapitre LXXXVIII Couverture Arc/Info E00 (ASCII)

Les couvertures Arc/Info E00 (par exemple Arc/Info V7 et plus récent) sont gérées par OGR en lecture.

Les sections label, arc, polygone, centroïde, région et texte d'une couverture sont toutes gérées comme couches. Les attributs d'INFO sont ajoutés aux labels, arcs, polygones ou régions à l'endroit approprié. Lorsque cela est disponible les informations de projection sont lu et traduite. Les géométries polygonales sont collectées pour les couches polygonales et régions à partir des arcs de compositions.

Les sections textes sont représentées comme des couches de points. La hauteur d'affichage est préservée dans le champ attributaire HEIGHT ; cependant, les autres informations sur l'orientation du texte sont ignorées.

Les tables info associées à une couverture mais pas spécialement nommé pour être reliée à une des couches géométriques existantes n'est pas actuellement accessible par OGR. Notez que les tables info sont stockées dans un répertoire 'info' au même niveau que le répertoire de couverture. Si cela est inaccessible ou corrompue aucun attribut info ne seront ajoutés aux couches de couverture, mais la géométrie devrait toujours être accessible.

Les couches sont nommé comme suit :

- une couche label (labels polygonaux, ou points libres) est nommée LAB si présente.
- une couche centroïde (centroïde de polygone) est nommée CNT si présente.
- une couche arc (ligne) est nommée ARC si présente.
- une couche polygone est nommée "PAL" si présente.
- une section texte est nommée en fonction de la sous-classe de la section.
- une sous-classe d'une région est nommée en fonction du nom de la sous-classe.

La lecture aléatoire (par FID) d'arcs ou de polygone est gérée, elle peut ne pas être gérée pour d'autres types d'objets. L'accès aléatoire aux fichiers E00 est généralement lente.

A Voir aussi

- Page de la bibliothèque AVCE00

- Pilote OGR pour AVCBin (Couverture binaire)

Chapitre LXXXIX BNA - Atlas BNA

Le format BNA est un format d'échange ASCII pour les données vecteurs 2D géré par plusieurs applications. Il contient seulement les géométries et quelques identifiants par enregistrement. Les attributs doivent être stockés dans des fichiers externes. Il ne gère pas les informations du système de coordonnées.

OGR gère la lecture et l'écriture au format BNA.

Le pilote OGR gère la lecture et l'écriture de tous les types de géométrie du format BNA :

- points
- polygones
- lignes
- ellipses/cercles

Comme le format BNA n'a pas de spécification formelle, il peut y avoir plusieurs formes de fichiers de données BNA. Le pilote OGR fait de son mieux pour lire les jeux de données BNA et gérer les formats d'enregistrements en simple ou multi-ligne, les enregistrements avec 2, 3 ou 4 identifiants, etc. Si vous avez un fichier de données BNA qui ne peut être lu correctement par le pilote BNA, s'il vous plait créer un rapport de bug sur le système de suivi de GDAL.

Pour être reconnue comme BNA, l'extension du fichier doit être ".bna". Lors de la lecture d'un fichier BNA, le pilote le scannera entièrement pour retrouver quelles couches sont disponibles. Si le nom du fichier est foo.bna, les couches seront nommées *foo_points*, *foo_polygons*, *foo_lines* et *foo_ellipses*.

Le pilote BNA gère la lecture des polygones avec des trous ou des iles. Il détermine ce qui est un trou ou une île seulement par analyse géométrique (tests d'inclusion, de non-intersection) et ignore complètement la notion de polygone "sinueux" (polygon winding) (si les bords d'un polygone sont décrits dans le sens des aiguilles d'une montre ou à l'inverse). GDAL doit être compilé avec GEOS pour permettre ces tests. Les polygones sont présentés comme des multipolygones dans le modèle Simple Feature d'OGR.

Les ellipses et cercles sont transformés en polygone avec 360 points.

A Problèmes lors de la création

Lors de l'export, toutes les couches sont écrites dans un seul fichier BNA. La mise à jour

de fichiers existants n'est pas gérée pour l'instant.

Si le fichier en sortie existe déjà, l'écriture n'aura pas lieu. Vous devez supprimer le fichier existant d'abord.

Le créateur BNA gère les options de création suivantes (options de jeu de données) :

- **LINEFORMAT :** par défaut, lors de la création de nouveaux fichiers ceux-ci sont créés avec la convention de fin de ligne de la plate-forme local (CR/LF sur win32 ou LF on sur tous les autres systèmes). Cela peut être écrasée par l'utilisation de l'option de création de couche *LINEFORMAT* qui peut avoir une valeur CRLF (format DOS) ou LF (format Unix).
- **MULTILINE :** par défaut, les fichier BNA sont créés au format multiligne (pour chaque enregistrement, la première ligne contient les identifiants et le type/nombre de coordonnées à suivre. Les lignes suivantes contiennent une paire de coordonnées). Cela peut être écrasée à travers l'utilisation de *MULTILINE=NO*.
- **NB_IDS :** les enregistrements BNA peuvent contenir de 2 à 4 identifiants par enregistrement. Certaines applications gère seulement un nombre d'identifiants précis. Vous pouvez écraser la valeur par défaut (2) par une valeur précise : 2,3 ou 4, ou *NB_SOURCE_FIELDS*. *NB_SOURCE_FIELDS* signifie que le fichier en sortie contiendra le même nombre d'identifiants que les objets écrits (fixé entre 2 et 4).
- **ELLIPSES_AS_ELLIPSES :** Le *writer* BNA tentera de reconnaitre les ellipses et les cercles lors de l'écriture des polygones. Cela fonctionnera seulement si l'objet a été précédemment lu à partir d'un fichier BNA. Comme certaine application ne gère pas les cercles/ellipses dans un fichier de données BNA, il peut être utile de demander au *writer* en spécifiant *ELLIPSES_AS_ELLIPSES=NO* de ne pas les exporter tel quel, mais de les laisser sous forme de polygones.
- **NB_PAIRS_PER_LINE :** cette option peut être utilisée pour limiter le nombre de paires de coordonnées par ligne dans le format multiligne.
- **COORDINATE_PRECISION :** cette option peut être utilisée pour définir le nombre de décimal pour les coordonnés. 10 par défaut.

B Exemple

La commande ''ogrinfo'' peut être utilisée pour faire un dump du contenu des fichiers de données BNA :

```
ogrinfo -ro -al a_bna_file.bna
```

La commande ''ogr2ogr'' peut être utilisée pour réaliser une traduction du format BNA vers le format BNA :

```
ogr2ogr -f BNA -dsco "NB_IDS=2" -dsco "ELLIPSES_AS_ELLIPSES=NO"
output.bna input.bna
```

C Voir également

- Description du format de fichier BNA
- Une autre description du format de fichier BNA
- Archive des produits lié à Census (ACRP) : téléchargement de jeu de données BNA de fichier de limite basé sur des fichiers TIGER 1992 contenant les

géographies des États-Unis

Chapitre XC CouchDB - CouchDB/GeoCouch

(GDAL/OGR >= 1.9.0)

Ce pilote peut se connecter à un service CouchDB, potentiellement avec l'extension spatial GeoCouch. GDAL/OGR doit avoir été compilé avec la gestion de Curl pour compiler le pilote de CouchDB.

Le pilote gère les opérations en lecture et écriture.

A Concepts CouchDB vs OGR

Une base de données CouchDB est considérée comme un couche OGR. Un document CouchDB est considéré comme une feature OGR.

OGR prend en charge de préférence les documents CouchDB qui suivent les spécifications GeoJSON.

B Syntaxe du nom de jeu de données

La syntaxe pour ouvrir une source de données CouchDB est : `couchdb:http://example.com[/layername]` où http://example.com pointe vers la racine d'un dépôt CouchDB et, en option, *layername* est le nom de la base de données CouchDB.

Il est également possible d'ouvrir directement une vue : `couchdb:http://example.com/layername/_design/adesigndoc/_view/aview[?include_docs=true]` *include_docs=true* peut être nécessaire en fonction de la valeur renvoyée par l'appel de *emit()* dans la fonction *map()*.

C Authentification

Certaines opérations, en particulier les opérations d'écriture, nécessitent une authentification. L'authentification peut être envoyée avec la variable d'environnement *COUCHDB_USERPWD* définie par *user:password* ou directement dans l'URL.

D Filtrage

Le pilote transmettra n'importe quel filtre défini avec *SetSpatialFilter()* vers le serveur quand l'extension GeoCouch est disponible. Il fera de même pour les filtres attributaires (très simples) définis avec *SetAttributeFilter()*. Lorsque l'évaluation des filtres échoue côté serveur, le filtre sera évalué côté client.

Par défaut, le pilote tentera d'utiliser la fonction filtre spatial suivante "_design/ogr_spatial/_spatial/spatial", qui est la fonction filtre spatial valide pour les couches créées par OGR. Si cette fonction filtre n'existe pas, mais qu'une autre existe, vous pouvez la définir avec l'option de configuration *COUCHDB_SPATIAL_FILTER*.

Notez que la première fois qu'une requête d'attribut est envoyée, il peut être nécessaire d'avoir des permissions d'écriture dans la base de données pour créer une nouvelle vue d'index.

E Pagination

Les features sont récupérées à partir du serveur par tranche de 500 par défaut. Ce nombre peut être modifié avec l'option de configuration *COUCHDB_PAGE_SIZE*.

F Gestion de l'écriture

La création et la suppression de table sont possibles.

La gestion de l'écriture est seulement activée lorsque la source de données est ouverte en mode update.

Lors de l'insertion d'une nouvelle feature avec *CreateFeature()*, et si la commande réussit, OGR ira chercher les champs _id et _rev renvoyés et les utilisera.

G Gestion de l'écriture et transactions OGR

Les opérations *CreateFeature()* / *SetFeature()* sont par défaut transmises au serveur en synchronisation avec l'appel de l'API d'OGR. Cela peut cependant entrainé des problèmes de performances lorsque de nombreuses commandes sont enclenchées à cause d'un surplus d'échanges clients/serveurs..

Il est possible d'entourer les opérations *CreateFeature()* / *SetFeature()* entre *OGRLayer::StartTransaction()* et *OGRLayer::CommitTransaction()*. Les opérations seront stockées en mémoire et seulement exécutées au moment de l'appel de *CommitTransaction()*.

H Options de création de couche

Les options de création de couche suivantes sont gérées :

- **UPDATE_PERMISSIONS = LOGGED_USER|ALL|ADMIN|function(...)| DEFAULT :** Met à jour les permissions pour la nouvelle couche.
  - si définie à LOGGED_USER (défaut), seuls les utilisateurs identifiés pourront réaliser des modifications sur la couche.
  - si définie à ALL, tous les utilisateurs pourront faire des modifications sur la couche.
  - si définie à ADMIN, seulement les administrateurs pourront faire des

modifications sur la couches.
    - si elle commence par "function(", la valeur de l'option de création sera utilisée comme contenu de la fonction validate_doc_update.
    - sinon, tous les utilisateurs seront autorisés à réaliser des modifications dans les *non-design* documents.
- **GEOJSON = YES|NO :** définissez la à *NO* pour éviter l'écriture de documents comme documents JSON. *Yes* par défaut.
- **COORDINATE_PRECISION = int_number :** nombre maximal de chiffre après le séparateur décimal à écrire pour les coordonnées. 15 par défaut. Une troncature intelligente sera réalisée pour supprimer les zéros inutiles.

Note!

Lors de l'ouverture d'un jeu de données en mode update, l'option de configuration OGR_COUCHDB_COORDINATE_PRECISION peut être définie pour avoir un rôle similaire.

I Exemples

Lister les tables d'un répertoire CouchDB :

```
ogrinfo -ro "couchdb:http://some_account.some_couchdb_server.com"
```

Créer et remplir une table à partir d'un shapefile :

```
ogr2ogr -f couchdb
"couchdb:http://some_account.some_couchdb_server.com" shapefile.shp
```

J Voir également

- Référence de CouchDB
- Dépôt du code source de GeoCouch
- Documentation pour la fonction 'validate_doc_update'

Chapitre XCI Comma Separated Value (.csv)

OGR gère la lecture et l'écriture de données essentiellement tabulaire non-spatiale dans des fichiers CSV texte. Les fichiers CSV sont des formats d'échange commun entre les logiciels gérant les données tabulaires et sont également facilement produit manuellement avec un éditeur de texte ou avec un programme ou des scripts écrit par l'utilisateur.

Bien qu'en théorie les fichiers csv peuvent avoir n'importe quelle extension, dans le but de reconnaitre automatiquement le format OGR gère seulement les fichiers se terminant avec l'extension csv. Le nom de la source de données peut être soit un fichier seul ou un répertoire. Pour qu'un répertoire soit reconnu comme une source de données csv au moins la moitié des fichiers dans le répertoire doivent avoir l'extension csv. Une couche (table) est produite pour chaque fichiers csv accédés.

Starting with GDAL 1.8.0, for files structured as CSV, but not ending with .CSV extension, the 'CSV:' prefix can be added before the filename to force loading by the CSV driver.

Le pilote CSV d'OGR gère la lecture et l'écriture. Parce que le format CSV a des lignes de longueur variable, la lecture est réalisée séquentiellement. La lecture des géométries dans un ordre aléatoire sera généralement très lente. Les couches CSV d'OGR n'ont jamais de systèmes de coordonnées pour les objets. Lors de la lecture d'un champ nommé "WKT" il est supposé contenir une géométrie WKT, mais est également traité comme un champ normal. Le pilote CSV d'OGR renverra tous les attributs des colonnes avec un type chaine de caractère s'il n'y a aucun fichier d'information sur le type des champs (avec l'extension .csvt) n'est disponible.

Une reconnaissance limitée des types peut être réalisé pour les entiers (Integer), les réels (Real), les chaines de caractères (String), les dates (Date : YYYY-MM-DD), Time (HH:MM:SS+nn) et DateTime (YYYY-MM-DD HH:MM:SS+nn) par un fichier descriptif de même nom que le fichier CSV, mais avec l'extension .csvt. Les types de chaque colonne doivent être listés en une seule ligne : entouré de guillemet double et séparé par des virgules (par exemple, "Integer", "String"). Il est également possible de définir explicitement la largeur et la précision de chaque colonne,par exemple "Integer(5)","Real(10.7)","String(15)". Le pilote utilisera alors ces types comme spécifiés pour les colonnes CSV.

A Format

Les fichiers CSV ont une ligne pour chaque objet (enregistrement) dans la couche (table). Les valeurs du champ attributaire sont séparées par des virgules. Au moins deux champs par ligne doivent être présent. Les lignes peuvent se terminer par une terminaison de ligne du style DOS (CR/LF) ou Unix (LF). Chaque enregistrement doit avoir le même nombre de champs. À partir de GDAL 1.7.0, le pilote acceptera une virgule ou une tabulation comme séparateur de champs. Cette auto-détection fonctionnera seulement s'il n'y a pas d'autres séparateurs potentiels à la première ligne du ficheir CSV. Sinon la virgule sera la valeur par défaut comme séparateur.

Les valeurs des attributs complexes (tels que ceux contenant des virgules, des guillemets ou de nouvelles lignes) peuvent être placé entre double guillemet. N'importe quelle occurrence de guillemet double dans une chaine entre guillemet doit être doublé pour la protéger.

Le pilote tente de traiter la première ligne du fichier comme une liste de noms de champ pour tous les champs. Cependant, si un ou plusieurs champs sont numérique il est supposé que la première ligne est en réalité des valeurs de données et des noms de champs factices sont générés en interne ((field_1 à field_n) et le premier enregistrement est traité comme un objet.

Exemple (employee.csv):

```
ID,Salary,Name,Comments
132,55000.0,John Walker,"The ""big"" cheese."
133,11000.0,Jane Lake,Cleaning Staff
```

Notez que la valeur *Comments* pour le premier enregistrement de données est placé entre guillemet double à cause de la valeur contenant un guillemet, et ces guillemets doivent être doublé pour que nous sachions que nous avons pas atteint la fin de la chaine du guillemet.

Plusieurs variations des entrées textuel sont parfois appelées fichiers à Valeur Séparée par des Virgules (*Comma Separated Value*), incluant les fichiers sans virgule, mais à colonne à largeur fixe, ceux utilisant des tabulations comme séparateur ou ceux avec données auxiliaire définissant des types de champs ou de structure. Ce pilote ne tente pas de gérer de tel fichier mais à la place gère des fichiers .csv simple qui peuvent être reconnus automatiquement. Des scripts ou d'autres mécanismes peuvent généralement convertir les autres variations sous une forme qui est compatible avec le pilote CSV d'OGR.

B Lecture de fichier CSV contenant des informations spatiales

Il est possible d'extraire l'information spatiale (points) d'un fichier CSV qui possède des colonnes pour les coordonnées X et Y, par l'utilisation du pilote VRT.

Considérez le fichier CSV suivant (test.csv) :

```
Latitude,Longitude,Name
48.1,0.25,"First point"
49.2,1.1,"Second point"
47.5,0.75,"Third point"
```

Vous pouvez écrire le fichier VRT associé (test.vrt) :

```
<OGRVRTDataSource>
    <OGRVRTLayer name="test">
        <SrcDataSource>test.csv</SrcDataSource>
        <GeometryType>wkbPoint</GeometryType>
        <LayerSRS>WGS84</LayerSRS>
        <GeometryField encoding="PointFromColumns" x="Longitude"
y="Latitude"/>
    </OGRVRTLayer>
</OGRVRTDataSource>
```

et `ogrinfo -ro -al test.vrt` renverra :

```
OGRFeature(test):1
    Latitude (String) = 48.1
    Longitude (String) = 0.25
    Name (String) = First point
    POINT (0.25 48.1 0)

OGRFeature(test):2
    Latitude (String) = 49.2
    Longitude (String) = 1.1
    Name (String) = Second point
    POINT (1.1 49.200000000000003 0)

OGRFeature(test):3
    Latitude (String) = 47.5
    Longitude (String) = 0.75
    Name (String) = Third point
    POINT (0.75 47.5 0)
```

C Problèmes lors de la création

Le pilote gère la création de nouvelles base de données (comme un répertoire de fichier .csv), en ajoutant de nouveaux fichiers csv à un répertoire existant un fichier csv ou en ajoutant des objets à une table CSV existante. La suppression ou le remplacement d'objets existants n'est pas gérés.

Options de création de couche:

- **LINEFORMAT :** par défaut lors de la création d'un nouveau fichier csv ceux-ci sont créés avec les conventions de fin de ligne de la plateforme local (CR/LF sous win32 ou LF sur tous les autres systèmes). cela peut être écrasé par l'utilisation de l'option de création de couche *LINEFORMAT* qui peut avoir les valeurs *CRLF* (format DOS) ou *LF* (format Unix).
- **GEOMETRY (débute avec GDAL 1.6.0) :** par défaut, la géométrie d'un objet écrit dans un fichier csv est ignoré. Il est possible d'exporter la géométrie dans sa représentation WKT en spécifiant `GEOMETRY=AS_WKT`. Il est également posible d'exporter les géométries ponctuelles dans leurs composants X,Y,Z (différentes colonnes dans le fichier csv) en spécifiant `GEOMETRY=AS_XYZ`, `GEOMETRY=AS_XY` ou *GEOMETRY=AS_YX*. Les colonnes géométriques seront ajoutées à la colonne avec les valeurs des attributs.
- **CREATE_CSVT=YES/NO (débute avec GDAL 1.7.0) :** créer le fichier associé

.csvt (voir plus haut dans le paragraphe) pour décrire le type de chaque colonne de la cuoche et ses largeurs et précisions optionnelles. Valeur par défaut : NO

- **SEPARATOR=COMMA/SEMICOLON/TAB (à partir de GDAL 1.7.0):** caractère de séparateur de champ. Valeur par défaut : COMMA

1 Exemples

- cet exemple montre l'utilisation d``'ogr2ogr`` pour transformer un shapefile avec une géométrie ponctuelle en un fichier .csv avec les coordonnées X,Y,Z des points comme premières colonnes dans le fichier .csv

```
ogr2ogr -f CSV output.csv input.shp -lco GEOMETRY=AS_XYZ
```

D Sources de données particulières

Le pilote CSV peut également lire des fichiers dont la structure est proche des fichiers CSV :

- Fichier données Airport NfdcFacilities.xls, NfdcRunways.xls, NfdcRemarks.xls et NfdcSchedules.xls trouve sur le FAA website (OGR >= 1.8.0)
- Fichier du USGS GNIS (Geographic Names Information System) (OGR >= 1.9.0)
- The allCountries file from GeoNames (OGR >= 1.9.0 for direct import)

E Autres remarques

- le développement du pilote CSV d'OGR a été financé par DM Solutions Group et GoMOOS.

Chapitre XCII Microstation DGN

Les fichiers DGN de Microstation des versions antérieures à 8.0 de Microstation sont gérés en lecture. Le fichier complet est représenté comme une couche (nommé "élément").

Les fichiers DGN sont considérés avoir des informations de géoréférencement à travers OGR. Les objets possèderont les attributs générique suivants :

- **Type :** le code *integer* du type comme listé ci-dessous dans les éléments gérés.
- **Level :** le numéro de niveau de DGN (0-63).
- **GraphicGroup :** le numéro de groupe graphique.
- **ColorIndex :** l'index de la couleur dans la palette DGN.
- **Weight :** le poids du dessin (minceur) pour l'élément.
- **Style :** la valeur du style pour l'élément.

Les fichiers DGN ne contiennent pas d'indexes spatiaux ; cependant, le pilote DGN utilise les informations d'étendues au début de chaque élément pour minimiser le traitement des éléments en dehors de la fenêtre du filtre spatial courant quand cela est possible.

A Éléments gérés

Les types d'élément suivants sont gérés :

- **Line (3) :** géométrie linéaire ;
- **Line String (4) :** géométrie linéaire multi-segment ;
- **Shape (6) :** géométrie polygonale ;
- **Curve (11) :** approximé comme une géométrie linéaire ;
- **B-Spline (21) :** traité (avec inexactitude) comme une géométrie linéaire ;
- **Arc (16) :** approximé comme une géométrie linéaire ;
- **Ellipse (15) :** approximé comme une géométrie linéaire ;
- **Text (17) :** traité comme une géométrie ponctuelle

De manière générale tout concept d'objets complexes, et de cellules comme composant associé est perdu. Chaque composant d'un objet complexe ou d'une cellule est traité comme un objet indépendant.

B Information de style

Certaine information de dessin sur les objets peuvent être extraite à partir des attributs génériques *ColorIndex*, *Weight* et *Style* ; cependant pour tous les objets, une chaine style d'OGR a été préparé avec les valeurs encodées sous une forme prête à être utilisée pour les applications utilisant les chaines style d'OGR.

Les différents sortes de géométries linéaires apporteront des informations de style indiquant la couleur, l'épaisseur et le style de la ligne (c'est à dire pointillé, solide etc.).

Les polygones (éléments Shape) apporteront les informations de style pour les bords ainsi que une couleur de remplissage si elle est fournie. Les motifs de remplissage ne sont pas gérés.

Les éléments textuels contiendront les informations sur le texte, l'angle, la couleur et la taille (exprimé en unité de groupe) dans la chaine style.

C Options de création

Les fichiers DGN 2D peuvent être écrit avec OGR avec des limitations significatives :

- les objets en sortie ont les attributs usuel DGN fixés. Toutes tentatives de créer de nouveau champ échouera.
- Quasiment aucun effort n'est pour l'instant réalisé pour traduire les chaines styles des objet OGR en information de représentation DGN.
- Les géométries POINT qui n'ont pas de texte (le texte est NULL, et la chaine style de l'objet n'est pas un LABEL) sera traduit comme élément ligne dégénéré (longueur 0).
- Les objets polygone, et multi-polygone seront traduit en simples polygones avec tous les anneaux après le premier omis.
- Les chaines polygones et ligne avec trop de sommet seront découpées en groupe d'éléments préfixé avec un élément *Complex Shape Header* ou *Complex Chain Header* comme approprié.
- Un fichier d'ensemencement doit être fournit (ou s'il ne l'est pas, *$PREFIX/share/gdal/seed_2d.dgn* sera utilisé). Plusieurs aspects du fichier DGN résultant sont déterminés par ce fichier, et ne peut être affecté par OGR, tel que la fenêtre de vue initiale.
- Les diverses géométries collection autre que *MultiPolygon* sont complètement ignoré pour l'instant.
- Les géométries qui tombent en dehors du "*design plane*" du fichier d'ensemencement seront ignorées, ou corrompu d'une manière imprévisible.
- Les fichiers DGN peuvent seulement avoir une couche. Toutes tentatives pour créer plus d'une couche dans nue fichier DGN échouera.

La création de jeu de donnés gère les options suivantes :

- **3D=YES or NO :** détermine si les fichiers d'ensemencement 2D (*seed_2d.dgn*) ou 3D (*seed_3D.dgn*)) doivent être utilisés. Cette option est ignorée si l'option *SEED* est fournie.
- **SEED=filename :** écrase le fichier d'ensemencement à utiliser.
- **COPY_WHOLE_SEED_FILE=YES/NO :** indique si tout le fichier d'ensemencement doit être copié. Si non, seulement les trois premiers éléments (et éventuellement la table de couleur) seront copiés. *NO* par défaut.
- **COPY_SEED_FILE_COLOR_TABLEE=YES/NO :** indique si la table de couleur doit être copiée à partir du fichier d'encemensement. Par défaut elle est définie à *NO*.

- **MASTER_UNIT_NAME=name :** écrase le nom de l'unité maitre à partir du ficher d'ensemencement avec celui d'un ou de deux caractères fournit.
- **SUB_UNIT_NAME=name :** écrase le nom de la sous unité à patir du fichier d'ensemencement avec celui d'un ou de deux caractères fournit.
- **SUB_UNITS_PER_MASTER_UNIT=count :** écrase le nombre de sus unité par unité maitre. La valeur du fichier d'ensemencement est utilisé par défaut.
- **UOR_PER_SUB_UNIT=count :** écrase le nombre de UOR (Unité de Résolution, *Units of Resolution*) par sous unité. La valeur du fichier d'ensemencement est utilisée par défaut.
- **ORIGIN=x,y,z :** écrase l'origine du *design plane*. L'origine du fichier d'ensemencement est utilisée par défaut.

D Voir également

- Page Dgnlib
- Spécification des Styles des objets d'OGR

Chapitre XCIII DODS/OPeNDAP

Ce pilote implémente la gestion en lecture seule pour la lecture des données géométriques à partir de serveurs OPeNDAP (DODS). Il est inclus en option dans OGR s'il est compilé avec la bibliothèque de gestion d'OPeNDAP.

Lors de l'ouverture d'une base de données, son nom doit être définie sous la forme *DODS:url*. L'URL peut inclure une expression de contrainte comme vue ici. Notez qu'il peut être nécessaire de mettre entre guillemet ou tout autre protection de l'URL DODS dans la ligne de commande si elle inclue des points d'interrogation ou des esperluettes (&) puisqu'ils ont souvent des significations dans une console.

```
DODS:http://dods.gso.uri.edu/dods-3.4/nph-dods/broad1999?&press=148
```

Par défaut, les Séquences, Grilles et Objets tableau de haut niveau seront traduit en couche correspondante. Les Séquences sont (par défaut) traitées comme des couches ponctuelles avec la géométrie récupéré à partir des variables lat et lon si disponible. Pour fournir une traduction plus sophistiquée des séquences, grilles et tableaux vers des géométries il est nécessaire de fournir des informations additionnelles à OGR sous forme de DAS (*dataset auxilary information*) soit sous forme de serveur distant soit sous forme locale via le mécanisme AIS.

Une définition DAS pour une couche OGR doit ressembler à ceci :

```
Attributes {
    ogr_layer_info {
        string layer_name WaterQuality;
        string spatial_ref WGS84;
        string target_container Water_Quality;
        layer_extents {
            Float64 x_min -180;
            Float64 y_min -90;
            Float64 x_max 180;
            Float64 y_max 90;
        }
        x_field {
            string name YR;
            string scope dds;
```

```
        }
        y_field {
            string name JD;
            string scope dds;
        }
    }
}
```

A Avertissement

- Aucune largeurs de champ n'est capturée pour les champs attributaires à partir du format DODS.
- Les performances pour des requêtes répétées sont améliorées de façon significative en activant le cache du format DODS. Essayez la définition de *USE_CACHE=1* dans votre fichier *~/.dodsrc*.

B Voir également

- OPeNDAP

Chapitre XCIV DXF d'AutoCAD

OGR gère la lecture de la plupart des versions DXF d'AutoCAD et l'écriture des fichiers de versions AutoCAD 2000. DXF est un format ascii utilisé pour l'échange de données AutoCAD entre différents logiciels de l'éditeur. Le contenu complet du fichier est représenté comme un couche unique nommée "entities".

Les fichiers DXF sont considéré sans informations de géo-référencement via OGR. Les features possèderont toutes les attributs génériques suivants :

- Layer : le nom de la couche DXF. Par défaut "0".
- SubClasses : si disponible, une liste de classes à laquelle un élément appartient.
- ExtendedEntity : si disponible, attributs de l'entité étendue toute empilée pour former un attribut texte unique.
- Linetype : si disponible, le type de ligne utilisée pour cette entité.
- EntityHandle : de quoi gérer l'entité en hexadecimal. Une sorte d'identifiant de la feature.
- Text : le texte des étiquettes.

A Éléments gérés

Les types d'éléments suivants sont gérés :

- POINT : produit une feature de géométrie de type point.
- MTEXT, TEXT : produit une feature de géométrie de type point avec des informations de style de LABEL.
- LINE, POLYLINE, LWPOLYLINE : traduit en LINESTRING ou POLYGON s'il est fermé ou non. Les polylignes arrondis (ceux avec leur sommet déplacé d'attributs [1]) seront tesselés. Les polylignes à unique sommet sont traduit en POINT.
- CIRCLE, ELLIPSE, ARC : traduit en LINESTRING, en tesselant l'arc en segment de lignes.
- INSERT : une tentative est faite pour insérer la définition du block comme définie dans l'insert. Les blocks linework sont agrégés dans une seule feature avec une collection de géométrie. Les blocs de texte sont renvoyés comme un ou plusieurs features texte. Afin d'éviter la fusion de blocs dans une collection de géométrie l'option de configuration *DXF_MERGE_BLOCK_GEOMETRIES* peut être définie à *FALSE*.

1 [NdT] *those with their budge of vertices attributes set*

- DIMENSION : cet élément est découpé en feature en arrows et leaders, et une feature avec une étiquette dimension.
- HATCH : ligne et arc sont collectés comme des géométries polygone, mais aucun effort n'est entreprise actuellement pour représenter le style de remplissage d'entité HATCH.

Une tentative raisonnable est réalisée pour préserver la couleur, la largeur de la ligne, la taille et l'orientation du texte via les informations de styles des features lors de la traduction des éléments. Pour l'instant aucun effort n'est réalisé pour préserver les styles de remplissage ou les attributs de styles des lignes complexes.

L'approximation des arcs, ellipses, cercles et polylignes arrondis comme linestring est réalisée en découpant les arcs en sous-arcs inférieure au seuil d'un angle. Cet angle est définie par *OGR_ARC_STEPSIZE*. Par défaut à 4 degrés, mais peut être écrasé par la variable de configuration *OGR_ARC_STEPSIZE*.

B DXF_INLINE_BLOCKS

Le comportement par défaut pour les entités INSERT est d'être étendue avec la géométrie du bloc qu'ils référencent. Cependant si l'option de configuration *DXF_INLINE_BLOCKS* est définie à la valeur *FALSE*, alors le comportement est différent comme décrit ici.

- Une nouvelle couche nommée bloc sera disponible. Elle contiendra une ou plusieurs features, elle aura également un attribut *BlockName* qui indique quel bloc ils font partie.
- La couche entité aura de nouveau attributs BlockName, BlockScale, et BlockAngle.
- Les entités INSERT remplira ces nouveaux champs avec les informations correspondantes (ils sont null poru toutes les autres entités).
- Les entités INSERT ne contiendront pas de bloc géométrie - ils auront à la place une géométrie ponctuelle pour le point d'insertion.

L'intention est qu'avec le paramètre *DXF_INLINE_BLOCKS* désactivé, les blocs de références resteront comme références et le bloc original de définition sera disponible via la couche blocs. En export cette configuration entrainera la création de blocs identiques.

C Encodages des caractères

Normalement les fichiers DXF sont dans l'encodage ANSI_1252 / Win1252. GDAL/OGR tente de traduire cela vers l'UTF-8 lors de la lecture puis en ANSI_1252 pour l'écriture. Les fichiers DXF peuvent aussi avoir un champ en-tête ($DWGCODEPAGE) indiquant l'encodage du fichier. Dans GDAL 1.8.x et plus ancien cela était ignoré mais à partir de GDAL 1.9.0 et plus récent une tentative est réalisée pour utiliser cela pour re-encoder des autres encodages vers l'UTF-8. Cela fonctionnera en fonction du nom du code d'encodage et si GDAL/OGR a été compilé avec la bibliothèque iconv pour le re-encodage des caractères.

Dans certains as le paramètre *$DWGCODEPAGE* dans le fichier DXF sera erroné, ou ne sera pas reconnu par OGR. Il peut être édité manuellement, ou la variable de configuration *DXF_ENCODING* peut être utilisée pour écraser l'id qui sera utilisé par OGR lors du transcodage. La valeur de *DXF_ENCODING* doit être un nom d'encodage géré par *CPLRecode()* (ie un nom iconv), et pas un nom $DWGCODEPAGE du DXF. Utiliser le nom "UTF-8" pour le *DXF_ENCODING* évitera toute tentative de re-encodage

lors de la lecture.

D Problèmes de création

Les fichiers DXF sont écrits au format AutoCAD 2000. Un en-tête standard (Tout jusqu'au mot clé ENTITIES) est écrit à partir du fichier *$GDAL_DATA/header.dxf*, et le fichier *$GDAL_DATA/trailer.dxf* est ajouté après les entités. Une seule couche peut être créée dans le fichier output.

Les features points avec un style de LABEL sont écrit comme des entités MTEXT basé sur les informations de style.

Les features points sans style de LABEL sont écrits sous forme d'entité POINT.

LineString, MultiLineString, Polygon et MultiPolygons sont écrit sous forme d'une ou plusieurs entités LWPOLYLINE, fermé dans le cas d'un anneau polygonale. Un effort est réalisé pour préserver la largeur des lignes et leur couleur.

La création de jeu de données gère les options de création de jeu de données suivants :

- **HEADER=filename :** écrase le fichier d'en-tête utilisé - au lieu de header.dxf.
- **TRAILER=filename :** écrase le fichier trailer utilisé - au lieu de trailer.dxf.

Notez que dans GDAL 1.8 et supérieur, les modèles d'en-tête et trailer peut être des fichiers DXF complet. Le pilote les scannera et extraira seulement les portions nécessaires (portion avant ou après la section ENTITIES).

1 Références des blocs

Il est possible d'exporter un "bloc" de couche vers le DXF en plus de la couche "entities" dans le but de produire les définitions des BLOCKs réel DXF dans le fichier en sortie. Il est également possible d'écrire les entités INSERT si un nom de bloc est fournie pour une entité. Pour que cela fonctionne les conditions suivantes s'appliquent :

- une couche "blocks" peut être créée, et elle doit être créé avant la couche entité.
- les entités dans la couche blocks doivent avoir le champs BlockName remplis.
- Les objets à écrire comme INSERTs dans la couche entités doivent avoir une géométrie POINT et le champ BlockName définie.
- si un bloc (nom) est déjà définie dans l'en-tête modèle, celui-ci sera utilisé sans vérifier si une nouvelle définition a été fournie dans la couche block.

L'intention est qu'une simple traduction à partir d'un DXF avec *DXF_INLINE_BLOCKS* définie à FALSE reproduira approximativement les blocs originaux et gardera les entités INSERT comme des entités INSERT plutôt que des les éclater.

2 Définitions des couches

Lors de l'écriture des entités, si il est rempli le champs LayerName est utilisé pour définir la couche entités écrite. Si la couche n'est pas déjà définie dans le modèle d'en-tête alors une nouvelle définition de couche sera introduite, copiée de la définition de la couche par défaut ("0").

3 Définitions de type de ligne

Lors de l'écriture des entités LWPOLYLINE les règles suivantes s'appliquent au regard des définitions Linetype.

- Si le champ Linetype est définie sur les features écrites et que Linetype est déjà définie dans le modèle d'en-tête alors il sera référencé à partir des entités sans vérifier si un style OGR existe.
- Si le Linetype est définie mais que le Linetype n'est pas prédéfinie dans le modèle d'en-tête alors une définition sera ajoutée si la feature possède un style OGR avec un outil PEN et ne définition de motif "p".
- Si la feature n'a pas de champs Linetype définie, mais possède un style OGR avec un outil PEN avec un motif "p" définie alors un LineType automatiquement nommée sera créé dans le fichier en sortie.
- Il est supposé que les motifs utilisent les unités "g" (géoréférencé) pour définir le motif de la ligne. Sinon la mise à l'échelle des motifs DXF sera probablement fausse - potentiellement complètement fausse.

L'objectif est que le motif de style "dot dash" soient préservé lors de l'écriture vers le DFX et que les linetypes spécifiques puisse être prédéfinie dans le modèle d'en-tête et référencé en utilisant le champ Linetype si désiré.

Chapitre XCV EDIGEO

(GDAL/OGR >= 1.9.0)

Ce pilote lit les fichiers encodés dans le format d'échange français EDIGEO, un format de fichier texte dont l'objectif est l'échange d'informations géographiques entre SIG, avec de puissantes possibilité de description, modélisation topologique, etc.

Le pilote a été développé pour lire les fichiers du PCI (Plan Cadastral Informatisé) français produit par la DGI (Direction Générale des Impôts). Le pilote doit pouvoir aussi être capable d'ouvrir d'autres produits basé sur la norme EDIGEO.

Le fichier .THF décrivant l'échange EDIGEO doit être fournie au pilote et celui-ci lira les fichiers .DIC, .GEO, .SCD, .QAL and .VEC associés.

Dans l'objectif de définir correctement la projection des couches, le fichier IGNF qui contient la définition des SRS IGN doit être placé dans le répertoire des fichiers ressources PROJ.4.

L'ensemble des fichiers sera parsé en mémoire. Cela peut être une limitation s'il faut gérer un gros échange de fichier EDIGEO.

A Étiquettes

Pour les fichiers PCI d'EDIGEO, les étiquettes sont contenu dans la couche ID_S_OBJ_Z_1_2_2. OGR exportera les styles en suivant la spécification des styles des features d'OGR.

Il ajoutera également les champs suivants :

- **OGR_OBJ_LNK :** l'id du projet qui est lié à cette étiquette ;
- **OBJ_OBJ_LNK_LAYER :** le nom de la couche de l'objet lié ;
- **OGR_ATR_VAL :** la valeur de l'attribut à afficher (trouvé dans l'attribut ATR de l'objet *OGR_OBJ_LNK*) ;

- **OGR_ANGLE :** l'angle de rotation en degré (0 = horizontal, orienté dans le sens inverse des aiguilles d'une montre) ;
- **OGR_FONT_SIZE :** la valeur de l'attribut HEI multiplié par la valeur de l'option de configuration *OGR_EDIGEO_FONT_SIZE_FACTOR* dont la valeur par défaut est 2.

Combinés avec les attributs FON (font family), ils peuvent être utilisé pour définir le style dans QGIS par exemple.

Par défaut, OGR créera des couches spécifiques (xxx_LABEL) pour expédier en différente couche ID_S_OBJ_Z_1_2_2 en fonction de la valeur de xxx=OBJ_OBJ_LNK_LAYER. Cela peut être désactivé en définissant *OGR_EDIGEO_CREATE_LABEL_LAYERS* à NO.

B Voir également

- Introduction au standard EDIGEO (en Français)
- Standard EDIGEO - AFNOR NF Z 52000 (en Français)
- Standard d'échange des objets du PCI selon la norme EDIGEO (en Français)
- Page principale du Cadastre français (en Français)
- Description du module EDIGEO dans Geotools (en anglais)
- Échantillon de données EDIGEO

Chapitre XCVI Pilote de l'API FileGDB

A Aperçu

Le pilote FileGDB fournie un accès en lecture et écriture vers les sources de données basé sur FileGDB (celui créé par ArcGIS 10 et supérieur).

B Dépendances

- SDK de l'API FileGDB
- OGR >= 1.9.0

C Usage

Utilisé comme n'importe quel pilote basé sur des fichiers - pointez juste vers le répertoire FileGDB (un répertoire qui se termine avec le suffixe ".gdb").

- Lire à partir d'un FileGDB et charger dans PostGIS :

```
ogr2ogr -overwrite -skipfailures -f "PostgreSQL" PG:"host=myhost
user=myuser
  dbname=mydb password=mypass" "C:\somefolder\BigFileGDB.gdb"
"MyFeatureClass"
```

- Obtenir des infos détaillées du FileGDB :

```
ogrinfo -al "C:\somefolder\MyGDB.gdb"
```

D Notes de compilation

Lisez les exemple de compilation sous Windows de GDAL pour les plugins. Vous trouverez une section similaire dans nmake.opt pour FileGDB.

Après cela, allez dans le répertoire *$gdal_source_rootogrogrsf_frmtsfilegdb* et exécutez :

```
nmake /f makefile.vc plugin
```

```
nmake /f makefile.vc plugin-install
```

E Problèmes connus

Les champs date et blob n'ont pas été implémenté. C'est probablement juste quelques lignes de code, mais le développeur n'a pas eut assez de temps.

F Liens

- Page de l'API des fichiers de Geodatabase

Chapitre XCVII FMEObjects Gateway

Les sources d'objet gérées par FMEObjects sont gérés en lecture par OGR si le pont FMEObjects est configuré, et si une copie d'une licence de FMEObject est installée et accessible.

Pour utiliser les lecteurs basés sur FMEObjects le nom de la source de données passé doit être le nom du lecteur de FME à utiliser, deux points (:) puis le véritable nom de la source de donnés (c'est à dire le nom du fichier). Par exemple *NTF:F:DATANTF2144.NTF* indiquera que le lecteur NTF doit être utilisé pour lire le fichier. Il y a un certain nombre de cas spéciaux.

- une source de données se terminant par .fdd sera supposée être un fichier d'une <<Définition de source de données FME>> (*FME Datasource Definition*) qui contiendra le nom du lecteur, le nom de la source de données puis une définition de paire de ligne nom/valeur pour les macros qui conviennent pour passer l'appel à *createReader()*.
- Une source de données nommée *PROMPT* résultera à demander à l'utilisateur les informations en utilisant une boite de dialogue FME. Cela ne fonctionne que sous Windows.
- Une source de données nommée <<*PROMPT:filename*>> résultera à demander, puis d'avoir la définition résultante sauvée dans le fichier au format .fdd indiqué. L'extension .fdd sera ajouté au nom du fichier. Cela ne fonctionne que sous Windows.

Chaque objet FME sera traité comme une couche à traver OGR, nommé par le type de l'objet. Avec certaines limitations les systèmes de coordonnées FME sont gérés. Toutes les types de géométries de FME devraient être proprement gérés. Les attributs graphiques de FME (couleur, largeur de ligne, etc) ne sont pas convertie dans les informations de Style des objets d'OGR (OGR Feature Style).

A Cache

Dans le but d'activer l'accès rapide à de large jeux de données sans avoir à les traduire à chaque fois qu'ils sont accédés, le pont FMEObjects gère un mécanisme de cache d'objet lu à partir du lecteur FME dans un "Fast Feature Stores", un format vectoriel natif pour FME avec un index spatial pour la recherche spatiale rapide. Ces fichiers en cache sont gardés dans le répertoire indiqué par la variable d'environnement *OGRFME_TMPDIR* (ou

TMPDIR ou */tmp* ou *C:* si cela n'est pas disponible).

Le fichiers d'objet en cache aura un préfixe *FME_OLEDB_* et un index maitre est laissé dans le fichier *ogrfmeds.ind*. Pour enlever l'index supprimez tout ces fichiers. N'en enlevez pas juste quelques-uns.

Par défaut les objets dans le cache sont relus après 3 600 s (60 minutes). Le temps de rétention du cache est altéré au moment de la compilation en modifiant le fichier d'include *fme2ogr.h*.

Les entrées à partir des lecteurs de SDE et d'ORACLE ne sont pas en cache. Ces sources sont traitées spécialement de différentes manières également.

B Avertissements

- Établir une session FME est une opération assez dispendieuse sur un système Linux à 350Mhz, cela peut excéder 10 s.
- Les anciens fichiers dans le cache des objets sont nettoyés, mais seulement lors de visite ultérieure au code du pont de FMEObjects dans OGR. Cela signifie que même non utilisé, le pont FMEObjects laissera les anciens objets en cache indéfiniment.

C Configuration/compilation

Pour inclure un pont avec FMEObjects dans une compilation OGR il est nécessaire d'avoir chargé FME sur le système. Le paramètre de configuration *--with-fme=$FME_HOME* doit être fournie pour la configuration. Le pont FMEObjects n'est pas explicitement lié (il est chargé plus tard quand cela est nécessaire) il est donc pratique de distribuer un binaire compilé d'OGR avec la gestion de FMEObjects sans distribuer FMEObjects. Il fonctionnera seulement pour les personnes qui possèdent FMEObjects dans leur *path*.

Le pont FMEObjects a été testé sur Linux et Windows.

Plus d'information sur le produit FME et comment acheté une licence pour le logiciel FME (permettant la gestion de FMEObjects) peut être trouvé sur le site Internet de Safe Software à http://www.safe.com. Le développement de ce pilote a été financé par Safe Software.

Chapitre XCVIII GeoConcept Export (disponible à partir de GDAL 1.6.0)

Geoconcept est un SIG développé par la société GeoConcept SA. Il s'agit d'un SIG orienté-objet, dont les éléments sont nommés "objets", et le type d'éléments "type/sous-type" (les attributs sont transmissibles).

Le pilote OGR GeoConcept traite un fichier GeoConcept simple dans un dossier comme un jeu de données comprenant plusieurs couches. Actuellement, le pilote ne gère que les multi-polygones, les lignes et les points.

A Le format "fichier texte" de GeoConcept (GXT)

Parmi ses formats d'import/export, GeoConcept propose un format texte simple nommé .GXT (auparavant .TXT), qui peut contenir des objets de différents types/sous-types.

Les fichiers textes d'export GeoConcept devraient être disponibles en lecture et en écriture.

Les définitions de champs sont stockés dans un fichier .GCT associé, qui n'est utilisé que pour la création.

B Problèmes de création

Les fichiers GeoConcept peuvent contenir différentes sortes de géométrie (une par couche). Ceci rend très difficile la traduction d'une géométrie multiple d'un autre format vers GéoConcept avec ogr2ogr, car ce dernier ne permet pas de séparer les différents types de géométrie d'un fichier source.

Les sous-types sont considérés comme des éléments OGR. Le nom d'une couche est donc l'agrégation du **nom du type** de la couche, d'un **"."** et du **nom du sous-type** de la couche.

Les champs (fichier .GCT) connaissent un certain nombre de contraintes (TODO) :

- Les noms des attributs ne sont pas limités en longueur.
- Seuls les champs de types entier, réel et chaîne de caractère sont gérés. Les autres types ne peuvent pas être créés pour le moment, même s'ils existent dans

le modèle GeoConcept.

Le pilote OGR pour GeoConcept ne gère pas les fonctions de suppression.

C Options de création de jeux de données

EXTENSION=TXT|GXT : indique l'extension de l'export GeoConcept.

CONFIG=chemin_du_GCT : Dans le fichier GCT, chaque ligne doit commencer avec %%//#%% suivi par un mot-clé. Les lignes commençant par %%//%% sont des commentaires.

- Section de configuration : le fichier GCT commence avec %%//#SECTION CONFIG%% et finit avec %%//#ENDSECTION CONFIG%%. L'ensemble de la configuration est contenue entre ces marqueurs.
- Section Carte : uniquement pour documentation au moment de l'écriture de ce document. Cette section commence avec %%//#SECTION MAP%% et finit avec %%//#ENDSECTION MAP%%.
- Section Type : définit une classe de caractéristiques. Un type a un nom (Name) et un identifiant (ID). Un type contient des sous-types et des champs. Marqueurs : %%//#SECTION TYPE%% en début et %%//#ENDSECTION%% TYPE en fin.
- Section Sous-type : cette sous-section définit un type de caractéristique dans une classe. Un sous-type a un nom (Name), un identifiant (ID), un type de géométrie (Kind) et une dimension. Les types suivants de géométrie sont gérés : POINT, LINE et POLYGON. La version actuelle du pilote ne gère pas la géométrie TEXT. Les dimensions peuvent être 2D, 3DM ou 3D. Un sous-type contient les champs. Marqueurs : %%//#SECTION SUBTYPE%% et %%//#ENDSECTION SUBTYPE%%.
- Section Champs : Définit les champs de l'utilisateur. Un champ a un nom (Name), un identifiant (ID), un type (Kind). Les types suivants sont gérés : Entier, (INT), Réel (REAL), MEMO, Choix (CHOICE), Date, Heure (TIME), Longueur (LENGHT), Aire (AREA). Marqueurs : %%//#SECTION FIELD%% et %%//#ENDSECTION FIELD%%.

Les règles suivantes s'appliquent dans la section "champs" :

- Les champs aux noms privés commencent avec un @ (''Identifier, Class, Subclass, Name, NbFields, X, Y, XP, YP, Graphics, Angle'').
- Quelques champs privés sont obligatoires (ils doivent apparaître dans la configuration) : *Identifier*, *Class*, *Subclass*, *Name*, *X*, *Y*.
- Si le sous-type est linéaire (LINE), les champs suivants doivent être déclarés XP, YP.
- Si le sous-type est linéaire ou polygonal (LINE, POLYGON), ''Graphics'' doit être déclaré.
- Si le sous-type est ponctuel ou textuel (POINT, TEXT), ''Angle'' doit être déclaré. Quand cette option n'est pas utilisée,le pilote gère les types et sous-types soit sur la base du nom de la couche, soit en utilisant l'option ''-nln''.

D Options de création de couche

FEATURETYPE=TYPE.SUBTYPE : définit les éléments à créer. TYPE correspond à un des noms (Name) présents dans le fichier .GCT pour une section "type". SUBTYPE correspond au nom (Name) présent dans la section sous-type de la section type concernée.

At the present moment, coordinates are written with 2 decimales for cartesian +spatial

reference systems (including height) or with 9 decimales for +geographical spatial reference systems.

E Exemples

1 Exemple de fichier .GCT

```
//#SECTION CONFIG
//#SECTION MAP
//# Name=SCAN1000-TILES-LAMB93
//# Unit=m
//# Precision=1000
//#ENDSECTION MAP
//#SECTION TYPE
//# Name=TILE
//# ID=10
//#SECTION SUBTYPE
//# Name=TILE
//# ID=100
//# Kind=POLYGON
//# 3D=2D
//#SECTION FIELD
//# Name=IDSEL
//# ID=101
//# Kind=TEXT
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=NOM
//# ID=102
//# Kind=TEXT
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=WITHDATA
//# ID=103
//# Kind=INT
//#ENDSECTION FIELD
//#ENDSECTION SUBTYPE
//#ENDSECTION TYPE
//#SECTION FIELD
//# Name=@Identifier
//# ID=-1
//# Kind=INT
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@Class
//# ID=-2
//# Kind=CHOICE
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@Subclass
//# ID=-3
//# Kind=CHOICE
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@Name
```

```
//# ID=-4
//# Kind=TEXT
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@X
//# ID=-5
//# Kind=REAL
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@Y
//# ID=-6
//# Kind=REAL
//#ENDSECTION FIELD
//#SECTION FIELD
//# Name=@Graphics
//# ID=-7
//# Kind=REAL
//#ENDSECTION FIELD
//#ENDSECTION CONFIG
```

2 Exemple de fichier .GXT

```
//$DELIMITER "      "
//$QUOTED-TEXT "no"
//$CHARSET ANSI
//$UNIT Distance=m
//$FORMAT 2
//$SYSCOORD {Type: 2001}
//$FIELDS Class=TILE;Subclass=TILE;Kind=4;Fields=Private#Identifier
Private#Class   Private#Subclass        Private#Name
Private#NbFields        IDSEL   NOM     WITHDATA        Private#X
Private#Y       Private#Graphics
-1      TILE    TILE    TILE    3       000-2007-0050-7130-LAMB93
0       50000.00        7130000.00      4       600000.00
7130000.00      600000.00       6580000.00      50000.00
6580000.00      50000.00        7130000.00
+-1     TILE    TILE    TILE    3       000-2007-0595-7130-LAMB93
0       595000.00       7130000.00      4       1145000.00
7130000.00      1145000.00      6580000.00      595000.00
6580000.00      595000.00       7130000.00
+-1     TILE    TILE    TILE    3       000-2007-0595-6585-LAMB93
0       595000.00       6585000.00      4       1145000.00
6585000.00      1145000.00      6035000.00      595000.00
6035000.00      595000.00       6585000.00
+-1     TILE    TILE    TILE    3       000-2007-1145-6250-LAMB93
0       1145000.00      6250000.00      4       1265000.00
6250000.00      1265000.00      6030000.00      1145000.00
6030000.00      1145000.00      6250000.00
+-1     TILE    TILE    TILE    3       000-2007-0050-6585-LAMB93
0       50000.00        6585000.00      4       600000.00
6585000.00      600000.00       6035000.00      50000.00
6035000.00      50000.00        6585000.00
```

3 Exemple d'utilisation

Création d'un GXT :

```
ogr2ogr -f "Geoconcept" -a_srs "+init=IGNF:LAMB93" -dsco EXTENSION=txt
-dsco CONFIG=tile_schema.gct tile.gxt tile.shp -lco
FEATURETYPE=TILE.TILE
```

Annexer de nouveaux éléments à un fichier .GXT existant :

```
ogr2ogr -f "Geoconcept" -update -append tile.gxt tile.shp -nln
TILE.TILE
```

traduire un fichier .GXT en fichier Mapinfo :

```
ogr2ogr -f "MapInfo File" -dsco FORMAT=MIF tile.mif tile.gxt TILE.TILE
```

F Voir aussi

- Site officiel de GeoConcept

Chapitre XCIX GeoJSON

Ce pilote implémente la gestion en lecture et écriture pour l'accès aux géométries encodé au format GeoJSON. Le GeoJSON est un langage basé sur JavaScript Object Notation (JSON). Le JSON est un format léger textuel pour l'échange de données et GeoJSON n'est rien d'autre que sa spécialisation pour le contenu géographique.

Au moment d'écrire ce texte, GeoJSON est géré comme format de sortie de services implémenté par FeatureServer, GeoServer et CartoWeb.

Le pilote GeoJSON d'OGR traduit une donnée encodée en GeoJSON vers des objets du model Simple Feature d'OGR : jeu de données, couche, objet, géométrie. L'implémentation est basée sur le brouillon de spécification de GeoJSON, v5.0.

À partir de OGR 1.8.0, le pilote GeoJSON peut lire les sorties JSON des requêtes de services de Feature qui suivent les spécifications REST des GeoServices, comme implémenté par l'API du serveur REST d'ArcGIS.

A Source de données

Le pilote GeoJSON d'OGR accepte trois types de sources de données :

- *Uniform Resource Locator* (URL) - une adresse web pour réaliser des requêtes HTTP
- des fichiers textuels avec des données GeoJSON - identifié à partir de l'extension du fichier .geojson ou .json
- du texte passé directement et encodé en GeoJSON

B Couche

Un jeu de données GeoJSON est traduit à un objet *OGRLayer* simple avec un nom prédéfinie *OGRGeoJson* :

```
ogrinfo -ro http://featureserver/data/.geojson OGRGeoJSON
```

Il est également valide de faire l'hypothèse que *OGRDataSource::GetLayerCount()* pour la source de données GeoJSON retourne toujours 1.

Accéder un service Web comme source de données (par exemple FeatureServer), chaque

requête produira une nouvelle couche. Ce comportement se conforme à la nature *stateless* des transactions HTTP et est similaire à la façon dont opère les navigateurs : une requête = une page.

Si un membre de plus haut niveau des données GeoJSON est d'un autre type que FeatureCollection,le pilote produira une couche avec seulement un objet. Autrement, une couche consistera d'un ensemble d'objets.

C Objet

Le pilote GeoJSON d'OGR relie chaque objet des types suivants aux nouveaux objets *OGRFeature* : Point, LineString, Polygon, GeometryCollection, Feature.

Selon les spécification GeoJSON, seul l'objet *Feature* doit avoir un membre avec un nom de propriété. Chaque membre des propriétés est traduit vers un objet OGR du type de *OGRField* et ajouté à l'objet *OGRFeature* correspondant.

La spécification GeoJSON ne nécessite pas que tous les objets géométriques dans une colection doivent avoir le même schéma de propriétés. Si les objets géométriques dans un ensemble définie par un objet FeatureCollection ont différents schéma de propriétés, il en résulte alors un schéma de champs dans *OGRFeatureDefn* est généré comme l'union de toutes les propriétés géométriques.

Il est possible de dire au pilote de ne pas traiter les attributs en définissant la variable d'environnement *ATTRIBUTES_SKIP=YES*. Le comportement par défaut est de préserver tous les attributs (comme une union, voir paragraphe précédent), ce qui est équivalent à définir *ATTRIBUTES_SKIP=NO*.

D Géométrie

Comme pour le problème des objets avec des propriétés mixtes, le brouillon de la spécification GeoJSON ne nécessite pas que tous les objets géométriques dans une collection doivent avoir une géométrie de même type. Heureusement le modèle objet d'OGR permet d'avoir des géométries de plusieurs types dans une seule couche - un couche hétérogène. Par défaut, le pilote GeoJSON préserve le type de la géométrie.

Cependant, parfois il nécessite de générer une couche hétérogène à partir d'un ensemble d'objet géométrique hétérogène. Pour cela, il est possible de dire au pilote d'englober toutes les géométries avec un type *OGRGeometryCollection* comme un dénominateur commun. Ce comportement peut être contrôler par la variable d'environnement *GEOMETRY_AS_COLLECTION=YES* (NO par défaut).

1 Variables d'environnement

- *GEOMETRY_AS_COLLECTION* - utilisé pour contrôler la traduction des géométries : YES - englobe les géométries avec le type *OGRGeometryCollection*
- *ATTRIBUTES_SKIP* - contrôle la traduction des attributs : YES - ignore les attributs

E Option de création de couche

- **WRITE_BBOX = YES/NO :** (OGR >= 1.9.0) définie à YES pour écrire une propriété bbox avec la bounding box des géométries au niveau de la feature et de la collection de feature. NO par défaut.

- **COORDINATE_PRECISION = int_number :** (OGR >= 1.9.0) nombre maximal de chiffre à écrire après la virgule pour les coordonnées. 15 par défaut. Une coupure intelligente permettra de supprimer les zéros en trop.

F Exemple

Comment faire un dump du contenu d'un fichier .geojson :

```
ogrinfo -ro point.geojson
```

Comment réaliser une requête sur les objets à partir un service distant avec un filtre sur un attribut :

```
ogrinfo -ro http://featureserver/cities/.geojson OGRGeoJSON -where
"name=Warsaw"
```

Comment traduire un certain nombre d'objets à partir d'une requête d'un FeatureServer vers un shapefile d'ESRI :

```
ogr2ogr -f "ESRI Shapefile" cities.shp
http://featureserver/cities/.geojson OGRGeoJSON
```

Lire le résultat d'une requête FeatureService en fonction d'un serveur GeoServices REST :

```
ogrinfo -ro -al
  "http://sampleserver3.arcgisonline.com/ArcGIS/rest/services/Hydrogra
phy/Watershed173811/FeatureServer/0/query?where=objectid+
%3D+objectid&amp;outfields=*&amp;f=json"
```

G Lisez également

- GeoJSON - encoding geographic content in JSON
- JSON - JavaScript Object Notation
- JSON-C - Une implémentation JSON en C
- [Gdal-dev] OGR GeoJSON Driver - driver announcement
- Spécification REST des GeoServices

Chapitre C Base de données MDB de Geomedia

GDAL/OGR >= 1.9.0

OGR gère en option la lecture des fichiers .mdp de Geomedia via ODBC. Geomedia est une base de données Microsoft Access avec un ensemble de tables définies par Intergraph pour prendre en charge les métadonnées de la géodatabase, et des géométries pour la gestion des features dans une colonne BLOB dans un format particulier. Ce pilote accède à la base de données via ODBC mais ne dépend d'aucune couche middleware d'Intergraph.

Les fichiers .mdb de Geomedia sont accéder en passant le nom du fichier .mdb que l'on veut lire comme source de données. Sur Windows, aucun DSN ODBC n'est requis. Sous linux, il y a des problèmes avec les connexions sans DSN dû à une implémentation de cette fonctionnalité buguée ou incomplète dans l'outils MDB, il est donc nécessaire de configurer le Data Source Name (DSN) si le pilote de l'outil MDB est utilisé (vérifiez les instructions ci-dessous).

Dans le but de faciliter la compatibilité avec différentes configurations, l'option de configuration *GEOMEDIA_DRIVER_TEMPLATE* a été ajoutée pour fournir un moyen pour définir le DSN automatiquement avec le nom du fichier comme argument. Dans les cas où le nom du pilote est connu, cela permet la construction du DSN basé sur cette information d'une manière similaire à la valeur par défaut (utilisé pour l'accès Windows au pilote Microsoft Access).

OGR traite toutes les tables de features comme des couches. La plupart des types de géométries devrait être gérées (arcs ne l'est pas encore). Les informations du système de projection n'est pas géré pour l'instant.

Pour le moment le pilote de géodatabase Personnelle d'OGR ne tire aucun avantage des indexes spatiaux pour les requêtes spatiales rapides.

Par défaut, les commandes SQL sont envoyées directement au moteur de base de données MDB. Il est également possible d'interroger le pilote pour prendre en charge les commandes SQL avec le moteur :ref:`gdal.ogr.sql`, en passant la chaîne **"OGRSQL"** à la méthode *ExecuteSQL()* comme nom du dialect SQL.

Unknown interpreted text role "ref".

A Comment utiliser le pilote Geomedia avec unixODBC et les outils MDB (sous Unix et Linux)

À partir de GDAL/OGR 1.9.0, le pilote :ref:`gdal.ogr.formats.mdb` est une alternative pour la lecture des fichiers .mdb de Geomedia sans nécessité unixODBC et les outils MDB.

Unknown interpreted text role "ref".

Référez vous à la section similaire du pilote :ref:`gdal.ogr.formats.pgeo`. Le préfixe à utiliser pour ce pilote est *Geomedia:*.

Unknown interpreted text role "ref".

B Voir également

- :ref:`gdal.ogr.formats.mdb`
  Unknown interpreted text role "ref".

Chapitre CI GeoRSS : Geographically Encoded Objects pour les flux RSS

(Pilote disponible à partir de GDAL 1.7.0)

GeoRSS est une manière d'encoder une localisation dans des flux RSS ou Atom.

OGR gère la lecture et l'écriture du GeoRSS. La gestion de la lecture est seulement disponilbe si GDAL a été compilé avec la gestion de la bibliothèque expat.

Le pilote gère les documents RSS au format RSS 2.0 ou Atom 1.0.

Il gère également les 3 manières d'encoder la localisation : GeoRSS simple, GeoRSS GML et W3C Geo (ce dernier étant obsolète).

Le pilote peut lire et écrire des documents sans information de localisation également.

Le datum par défaut pour les documents GeoRSS est le datum WGS84 (EPSG:4326) bien que les localisations GeoRSS soit encodé dans l'ordre latitude-longitude dans le fichier XML, toutes les coordonnées reportées ou attendus par le pilote sont dans l'ordre longitude-latitude. L'ordre longitude/latitude utilisé par OGR est voulu pour compatibilité avec la plupart des autres pilotes et commandes OGR. Pour les localisations encodées dans GML, le pilote gérera l'attribut *srsName* pour la description des autres SRS.

L'encodage Simple et GML gèrent la notion de boîte comme une géométrie. Cela sera décodé comme un rectangle (géométrie polygonale) dans le modèle Simple Feature d'OGR.

Une couche seule est renvoyée pendant la lecture du document RSS. Les objets sont récupérés à partir du contenu des éléments <*item*> (document RSS) ou <*entry*> (document Atom).

A Problèmes d'encodage

La bibliothèque Expat gère la lecture des encodages internes suivants :

- US-ASCII
- UTF-8
- UTF-16
- ISO-8859-1

OGR 1.8.0 ajoute la gestion pour l'encodage Windows-1252 (pour les versions antérieures, la modification de l'encodage mentionné dans l'en-tête XML à ISO-8859-1 peut fonctionner dans certain cas).

Le contenu retourné par OGR sera encodé en UTF-8, après la conversion à partir de l'encodage mentionné par le fichier d'en-tête.

Si votre fichier GeoRSS n'est pas encodé dans un de ces encodages, il ne sera pas parsé par le pilote GeoRSS. Vous pouvez le convertir dans l'un des encodages gérés avec la commande *iconv* par exemple et changer en fonction la valeur du paramètre encodage dans l'en-tête XML.

Lors de l'écriture du fichier GeoRSS, le pilote s'attend à ce que du contenu en UTF-8 lui soit passé.

B Définitions des champs

Lors de la lecture d'un document, le pilote réalisera d'abord une lecture complète du document pour obtenir les définitions du champ.

Le pilote renverra les éléments trouvés dans le schéma de base d'un canal RSS ou d'un flux Atom. Il renverra également les éléments d'extension qui sont autorisés dans les espaces de nom.

Les attributs des éléments de premier niveau seront exposés comme champs.

Les contenus complexes (élément dans des éléments de premier niveau) seront renvoyé comme blob XML.

Quand un même élément est répété, un nombre apparaîtra à la fin du nom de l'attribut pour les répétitions. Cela est utile pour l'élément <*category*> dans les documents RSS et Atom par exemple.

Le contenu suivant :

```
<item>
    <title>Mon titre</title>
    <link>http://www.mylink.org</link>
    <description>Cool description !</description>
    <pubDate>Wed, 11 Jul 2007 15:39:21 GMT</pubDate>
    <guid>http://www.mylink.org/2007/07/11</guid>
    <category>Computer Science</category>
    <category>Logiciel Open Source</category>
    <georss:point>49 2</georss:point>
    <myns:name type="mon_type">Mon Nom</myns:name>
    <myns:complexcontent>
        <myns:subelement>Subelement</myns:subelement>
    </myns:complexcontent>
</item>
```

Sera interprété dans le modèle SF d'OGR comme :

```
title (String) = Mon titre
link (String) = http://www.mylink.org
description (String) = Cool description !
pubDate (DateTime) = 2007/07/11 15:39:21+00
guid (String) = http://www.mylink.org/2007/07/11
```

```
category (String) = Computer Science
category2 (String) = Logiciel Open Source
myns_name (String) = Mon Nom
myns_name_type (String) = mon_type
myns_complexcontent (String) =
<myns:subelement>Subelement</myns:subelement>
POINT (2 49)
```

C Problèmes lors de la création

À l'export, toutes les couches sont écrites en un seul fichier. La mise à jour de fichiers existant n'est pas gérée.

Si le fichier en sortie existe déjà, l'écriture n'aura pas lieu. Vous devez d'abord effacer le fichier existant.

Une couche qui est créé ne peut pas être immédiatement lu sans fermeture et réouverture du fichier. Ceci afin de dire qu'un jeu de données est en écriture seul ou en lecture seul au cours de la session.

Géométries gérées :

- Objets de type wkbPoint/wkbPoint25D.
- Objets de type wkbLineString/wkbLineString25D.
- Objets de type wkbPolygon/wkbPolygon25D.

Les autres types de géométrie ne sont pas gérés et seront ignoré silencieusement.

Le pilote GeoRSS gère ces options de création de jeux de données suivantes :

- **FORMAT=RSS|ATOM :** si le document doit être au format RSS 2.0 ou Atom 1.0 format. Valeur par défaut : RSS
- **GEOM_DIALECT=SIMPLE|GML|W3C_GEO (RSS or ATOM document) :** l'encodage des informations de localisation. Valeur par défaut : SIMPLE W3C_GEO ne gère que les géométries ponctuelles. SIMPLE ou W3C_GEO ne gère que les géométries avec des coordonnées WGS84.
- **USE_EXTENSIONS=YES|NO.*** valeur par défaut : *NO*. Si définie à *YES*, les champs étendus (c'est à dire le champs qui ne ont pas dans le schéma de base des documents RSS ou Atom) seront écrit. si le nom du champ non trouvé dans le schéma de base correspond au motif *foo_bar*, *foo* sera considéré comme le namespace de l'élément et un élément *<foo:bar>*sera écrit autrement les éléments seront écrit dans le namespace *<ogr:>*.
- **WRITE_HEADER_AND_FOOTER=YES|NO.*** valeur par défaut : *YES*. Si définie à *NO*, seul les éléments *<entry>* ou *<item>* seront écrit. L'utilisateur devra fournir les en-têtes et pieds de page appropriés du document. Les options suivantes ne sont pas utile dans ce cas.
- **HEADER (RSS ou document Atom) :** contenu XML qui sera placé entre l'élément *<channel>* et le premier élément *<item>*pour un document RSS, ou entre la balise xml et le premier élément *<entry>* pour un document Atom. Si cela est définie, cela écrasera les options qui suivent.
- **TITLE (RSS ou document Atom) :** les valeurs placées entre l'élément *<title>* dans l'en-tête. Si elle n'est pas fournie, une valeur factice sera utilisée puisque cet élément est obligatoire.
- **DESCRIPTION (document RSS) :** les valeurs placées entre l'élément *<description>* dans l'en-tête. Si elle n'est pas fournie, une valeur factice sera

utilisée puisque cet élément est obligatoire.

- **LINK (RSS document) :** les valeurs placées entre l'élément <*link*> dans l'en-tête. Si elle n'est pas fournie, une valeur factice sera utilisée puisque cet élément est obligatoire.
- **UPDATED (document Atom) :** les valeurs placées entre l'élément <*updated*> dans l'en-tête. Elle doit être formatée comme une datetime xml. Si elle n'est pas fournie, une valeur factice sera utilisée puisque cet élément est obligatoire.
- **AUTHOR_NAME (document Atom) :** les valeurs placées entre l'élément <*author*><*name*> dans l'en-tête. Si elle n'est pas fournie, une valeur factice sera utilisée puisque cet élément est obligatoire.
- **ID (document Atom) :** les valeurs placées entre l'élément <*id*> dans l'en-tête. Si elle n'est pas fournie, une valeur factice sera utilisée puisque cet élément est obligatoire.

Lors de la translation d'un de jeu de données source, il peut être nécessaire de renommer les noms du champ à partir du jeu de données source dans les noms d'attributs RSS ou ATOM attendu, tels que <*title*>, <*description*>, etc. Cela peut être réalisé avec un :ref:`gdal.ogr.formats.vrt`, ou en utilisant l'option *-sql* de la commande ogr2ogr (voir RFC21 : cast des types SQL d'OGR et alias des noms de champ)

Unknown interpreted text role "ref".

D Exemple

- la commande ''ogrinfo'' peut être utilisé pour dumper le contenu d'un fichier de données GeoRSS :

```
ogrinfo -ro -al input.xml
```

- la commande ''ogr2ogr'' peut être utilisé pour réaliser une translation de GeoRSS vers GeoRSS. Par exemple un document Atom dans un document RSS.

```
ogr2ogr -f GeoRSS output.xml input.xml "select link_href as
link, title,
  content as description, author_name as author, id as guid from
georss"
```

Note!

Dans cet exemple nous faisons une correspondance entre des champs équivalents à partir du nom source vers le nom attentdu du format de destination.

- Le script Python suivant montre comment lire le contenu d'un flux GeoRSS en ligne :

```
#!/usr/bin/python
import gdal
import ogr
import urllib2

url = 'http://earthquake.usgs.gov/eqcenter/catalogs/eqs7day-
M5.xml'
content = None
try:
    handle = urllib2.urlopen(url)
```

```
    content = handle.read()
except urllib2.HTTPError, e:
    print 'HTTP service for %s is down (HTTP Error: %d)' % (url,
e.code)
except:
    print 'HTTP service for %s is down.' %(url)

# Créé un fichier en mémoire à partir du contenu téléchargé
gdal.FileFromMemBuffer('/vsimem/temp', content)

ds = ogr.Open('/vsimem/temp')
lyr = ds.GetLayer(0)
feat = lyr.GetNextFeature()
while feat is not None:
    print feat.GetFieldAsString('title') + ' ' +
feat.GetGeometryRef().ExportToWkt()
    feat.Destroy()
    feat = lyr.GetNextFeature()

ds.Destroy()

# Libère la mémoire associé avec le fichier en mémoire
gdal.Unlink('/vsimem/temp')
```

E Voir aussi

- Page pour le format GeoRSS
- Page Wikipedia pour le format GeoRSS
- Page Wikipedia pour le format RSS
- Spécification RSS 2.0
- Page Wikipedia pour le format Atom
- Spécification Atom 1.0

Chapitre CII GFT - Google Fusion Tables

(GDAL/OGR >= 1.9.0)

Ce pilote peut se connecter au service Google Fusion Tables. GDAL/OGR doit être compilé avec la gestion de Curl pour que le pilote GFT soit compilé.

Le pilote gère la lecture et l'écriture des opérations.

A Syntaxe des noms des jeux de données

La syntaxe minimale pour ouvrir une source de données GFT est : `GFT:`

Des paramètres supplémentaires optionnels peuvent être définie après le signe ':' comme :

- **tables=table_id1[,table_id2] :** une liste des ID des tables. Cela est nécessaire lorsque vous devez accéder aux tables publiques. Si vous voulez l'ID d'une table publique, où n'importe quelle table qui n'appartient pas à l'utilisateur authentifié, vous devez allez voir la table dans le site Google Fusion Tables et noter le numéro à la fin de l'URL.
- **protocol=https :** pour utiliser le protocole HTTPS pour toutes les opérations. HTTP est utilisé par défaut sauf pour les opérations d'authentification où HTTPS est toujours utilisé.
- **email=martin.dupont@example.com :** l'email du compte Google utilisé pour authentification.
- **password=mot_de_passe_secret :** le mot de passe du compte Google utilisé pour l'authentification.
- **auth=auth_key :** une clé d'authentification comme décrit ci-dessous.

Si plusieurs paramètres sont définie, ils doivent être séparés par un espace.

B Authentification

La plupart des opérations, en particulier celles pour écrire, nécessite un compte Google valide pour fournir les informations d'authentification au pilote. La seule exception est l'accès en lecture seule des tables publiques.

Les informations d'authentification peuvent être fournie de différentes manières :

- en spécifiant les paramètres *email* et *mot de passe* dans le nom du jeu de données, comme décrit plus haut.
- en les spécifiant à travers les options de configurations/variables d'environnements *GFT_EMAIL* et *GFT_PASSWORD*.
- en spécifiant la clé authentification via l'option de configuration/ variable d'environnement *GFT_AUTH*. Cette valeur est une clé renvoyée par le serveur lors de l'utilisation de la méthode email + mot de passe, et qui peut être réutilisée pour une authentification ultérieure. Vous pouvez retrouver cette clé en lançant une opération ogrinfo préliminaire avec CPL_DEBUG définie à ON et en vous authentifiant avec une autre méthode. La valeur sera affichée après la chaîne "Auth key : ". Cette méthode possède l'avantage de ne pas exposer votre mot de passe en clair.
- en spécifiant la clé d'authentification via le paramètre de connexion *auth*.

C Géométrie

Géométries dans les tables GFT doivent être exprimées dans la projection géodésique WGS84. GFT les autorise d'être encodé sous différentes formes :

- une seule colonne avec une chaîne "lat lon" ou "lat,lon", où lat et lon sont exprimées en degré décimal.
- un seule colonne avec une chaîne KML qui est la représentation d'un Point, une LineString ou un Polygon.
- deux colonnes, une avec la latitude et l'autre avec la longitude, toutes deux exprimées en degré décimal.
- une seule colonne avec une adresse connu par le service de géocodage de Google Maps.

Seul les trois premiers types sont géré par OGR, pas le dernier.

Fusion tables peut avoir plusieurs colonnes géométriques par table. Par défaut, OGR utilisera la première colonne géométrique qu'il trouvera. Il est possible de sélectionner une autre colonne comme colonne géométrique en spécifiant *table_name(geometry_column_name)* comme nom de couche envoyé à *GetLayerByName()*.

D Filtre

Le pilote fera parvenir n'importe quel filtre spatial définie avec *SetSpatialFilter()* au serveur. Il fera de même pour les filtres attributaires définie via *SetAttributeFilter()*.

E Pagination

Les features sont récupérées à partir du serveur par tranche de 500 par défaut. Ce nombre peut être modifié avec l'option de configuration *GFT_PAGE_SIZE*.

F Gestion de l'écriture

La création et la suppression de table est possible. Notez que les champs ne peuvent qu'être ajouté à une table dans laquelle il n'y a pas de feature encore créé.

La gestion en écriture est seulement activée lorsque la source de données est ouverte en mode update.

La correspondance entre les opérations du service GFT et les concepts OGR est la

suivante :

- OGRFeature::CreateFeature() <==> INSERT operation
- OGRFeature::SetFeature() <==> UPDATE operation
- OGRFeature::DeleteFeature() <==> DELETE operation
- OGRDataSource::CreateLayer() <==> CREATE TABLE operation
- OGRDataSource::DeleteLayer() <==> DROP TABLE operation

Lors de l'insertion d'une nouvelle feature avec *CreateFeature()*, et si la commande est réussie, OGR récupérera le rowid renvoyé et l'utilisera comme FID. OGR reprojetera automatiquement ses géométries dans la projection géodésique WGS84 si nécessaire (si la projection originale est liée à la géométrie).

G Gestion de l'écriture et transactions OGR

Les opérations ci-dessus sont par défaut déclenchées vers le serveur synchrone avec l'appel à l'API d'OGR. Cela peut cependant causer des problèmes de performances lors de l'envoie de plusieurs commandes dû à de nombreux échanges client/serveur.

Il est possible d'encapsuler l'opération *CreateFeature()* entre *OGRLayer::StartTransaction()* et *OGRLayer::CommitTransaction()*. Les opérations seront stockées en mémoire et seulement exécutées lors de l'appel de *CommitTransaction()*. Notez que le service GFT gère seulement jusqu'à 500 INSERTs et jusqu'à 1 Mo de contenu par transaction.

Note!

Seule CreateFeature() active l'utilisation du mécanisme des transactions OGR. SetFeature() et DeleteFeature() seront toujours déclenchés immédiatement.

H SQL

Les commandes SQL envoyées aux appels *OGRDataSource::ExecuteSQL()* sont exécutées côté serveur, sauf si le dialecte OGRSQL est définie. Le sous ensemble de SQL géré par le service GFT est décrit dans le lien à la fin de cette page.

Le SQL géré par le serveur comprend seulement les id des tables et pas les noms des tables renvoyés par OGR. Pour convenance, cependant OGR modifiera vos commandes SQL pour remplacer le nom de la table par son id.

I Exemples

- Lister les tables et les vues de l'utilisateur authentifié :

```
ogrinfo -ro "GFT:email=john.doe@example.com
password=secret_password"
```

- Créer et peupler une table à partir d'un shapefile :

```
ogr2ogr -f GFT "GFT:email=john.doe@example.com
password=secret_password" shapefile.shp
```

- Afficher le contenu d'une table publique avec des filtres attributaires et spatiaux :

```
ogrinfo -ro "GFT:tables=224453" -al -spat 67 31.5 67.5 32 -where
"'Attack on' = 'ENEMY'"
```

- Obtenir la clé d'authentification :

```
ogrinfo --config CPL_DEBUG ON "GFT:email=john.doe@example.com
password=secret_password"
```

renvoie :

```
HTTP: Fetch(https://www.google.com/accounts/ClientLogin)
HTTP: These HTTP headers were set: Content-Type: application/x-
www-form-urlencoded
GFT: Auth key :
A_HUGE_STRING_WITH_ALPHANUMERIC_AND_SPECIAL_CHARACTERS
```

Maintenant vous pouvez définir la variable d'environnement GFT_AUTH à cette valeur et utiliser simplement "GFT:" comme DSN.

J Voir également

- Guide du développeur de Google Fusion Tables
- Référence des développeurs de Google Fusion Tables

Chapitre CIII GML - Geography Markup Language

OGR a une gestion limité pour la lecture et l'écriture du GML. La mise à jour de fichier existant n'est pas géré.

Version de GML gérées :

OGR version	*Read*	*Write*
OGR >= 1.8.0	GML2 et GML3 qui peut être traduit en modèle simple feature	GML 2.1.2 ou GML 3 SF-0 (GML 3.1.1 Compliance level SF-0)
OGR < 1.8.0	GML2 et GML3 limité	GML 2.1.2

A Parseur

La première fois qu'un fichier GML est ouvert il est complètement scanné dans le but d'obtenir l'ensemble des *featuretypes*, les attributs associés pour chacun d'eux et d'autres informations au niveau du jeu de données. Ces informations sont stockées dans un fichier .gfs avec le même nom que le fichier GML cible. Un accès ultérieur au même fichier GML utilisera ce fichier .gfs pour prédéfinir les informations de niveau du jeu de données accélérant son accès. Pour une étendues limité le fichier .gfs peut être édité manuellement pour modifier la manière dont le fichier GML sera parsé. Soyez avertie que le fichier .gfs sera automatiquement régénéré si le .gml associé a un timestamp supérieur.

Lors du pré-scan du fichier GML pour déterminer la liste des types d'objets, et les champs, les contenus des champs sont scannée pour tenter de déterminer le type du champ. Dans certaines applications cela est plus facile si tous les champs sont traité comme des champs de chaines de caractères. Cela peut être réalisé en définissant l'option de configuration *GML_FIELDTYPES* à la valeur *ALWAYS_STRING*.

OGR 1.8.0 ajoute la gestion pour détecter les attributs des feature dans les éléments GML imbriqués (hiérarchie d'attribut non plat) qui peut être trouvé dans certains profiles GML tels que ceux MasterMap de l'Ordnance Survey UK. OGR 1.8.0 apporte également la gestion de la lecture des champs de types IntegerList, RealList et StringList quand un

élément GML a plusieurs occurrences.

Depuis OGR 1.8.0, un pilote GML spécialisé - pilote :ref:`gdal.ogr.formats.nas` - est disponible pour lire le format d'échange GML AAA allemand (NAS/ALKIS).

Unknown interpreted text role "ref".

Les options de configuration peuvent être définie via la fonction *CPLSetConfigOption()* ou comme variables d'environnement.

B Lecture des géométries

Lors de la lecture d'une feature, le pilote prendra par défaut seulement en compte la dernière géométrie GML reconnu trouvée (dans le cas où il y en a plusieurs) dans le sous arbre XML décrivant la feature.

À partir de OGR 1.8.0, l'utilisateur peut changer le fichier .gfs pour sélectionner la géométrie appropriée en spécifiant son chemin avec l'élément <GeometryElementPath>. Voyez la description de la syntaxe .gfs plus bas.

OGR 1.8.0 ajoute la gestion de plusieurs géométries GML incluant TopoCurve, TopoSurface, MultiCurve. Le type géométrie GML TopoCurve peut être interprété comme l'un des deux types de géométries. Les éléments Edge interne contiennent des courbes et leurs noeuds correspondants. Par défaut seules les courbes, la géométrie principale, sont retournées comme OGRMultiLineString. Pour récupérer les noeuds, sous forme de OGRMultiPoint, l'option de configuration **GML_GET_SECONDARY_GEOM** doit être définie à la valeur **YES**. Lorsque cela est fait seul les géométries secondaires sont renvoyées.

C Résolution gml:xlink

OGR 1.8.0 ajoute la gestion de la résolution des gml:xlink. Quand le *résolveur* trouve un élément contenant une balise xlink:href, il tente de trouver l'élément correspondant avec le gml:id dans le même fichier gml, d'autre fichier gml dans le système de fichier ou sur le web en utilisant cURL. Définissez l'option de configuration **GML_SKIP_RESOLVE_ELEMS** à **NONE** pour activer la résolution.

Par défaut le fichier résolu sera sauvé dans le même répertoire que le fichier originel avec l'extension ".resolved.gml", s'il n'existe pas déjà. Ce comportement peut être changé en utilisant l'option de configuration **GML_SAVE_RESOLVED_TO**. Définie le à **SAME**

pour écraser le fichier original. Définissez le à **un nom de fichier finissant par .gml** pour le sauver à cet endroit. Toutes autres valeurs seront ignorées. Si le *résolveur* ne peut pas écrire dans le fichier pour n'importe quel raison, il tentera de le sauver dans un fichier temporaire généré par *CPLGenerateTempFilename("ResolvedGML");* sinon la résolution échouera.

Notez que l'algorithme de résolution n'est pas optimisé pour les gros fichiers. Pour les fichiers avec plus de 2000 balises xlink:href, le process peut durer plus que quelques minutes. Une progression approximative est affichée grâce à *CPLDebug()* tous les 256 liens. Il peut être vue en définissant la variable d'environnement CPL_DEBUG. Le temps de résolution peut être réduit si vous connaissez les éléments qui ne sont pas nécessaire. Mentionnez une liste de noms séparés par des virgules des éléments avec l'option de configuration **GML_SKIP_RESOLVE_ELEMS**. Définissez à **ALL** pour ne pas réaliser la résolution en même temps (défaut). Définissez à **NONE** pour résoudre tous les xlinks.

D Problèmes d'encodage

La bibliothèque Expat gère la lecture des encodages internes suivants :

- US-ASCII
- UTF-8
- UTF-16
- ISO-8859-1

Lorsqu'il est utilisé avec la bibliothèque Expat, OGR 1.8.0 ajoute la gestion de l'encodage Windows-1252 (pour les versions précédentes, modifier l'encodage mentionnée dans l'en-tête XML à ISO-8859-1 peut fonctionner dans certain cas).

Le contenu renvoyé par OGR sera encodé en UTF-8, après la conversion à partir de l'encodage mentionné dans l'en-tête du fichier.

Si le fichier GML n'est pas encodé dans l'un des encodages précédents et que le seul parseur disponible est Expat, il ne sera pas parsé par le pilote GML. Vous pouvez le convertir dans l'un des encodages gérés avec la commande *iconv* par exemple et changer la valeur du paramètre *encoding* dans l'en-tête XML en conséquence.

E Feature id (fid / gml:id)

À partir de OGR 1.9.0, le pilote expose le contenu de l'attribut *gml:id* comme champ de chaîne de caractères appelé *gml_id*, lors de la lecture des documents GML des WFS. Lors de la création d'un document GML3, si un champ est appelé *gml_id*, son contenu sera également utilisé pour écrire le contenue de l'attribut *gml:id* de la feature créée.

À partir de OGR 1.9.0, le pilote auto-détecte la présence d'un attribut fid (GML2) (resp. gml:id (GML3)) au début du fichier, et, si présent, l'expose par défaut en tant que champ *fid* (resp. *gml_id*). L'auto-détection peut être écrasée en spécifiant l'option de configuration **GML_EXPOSE_FID** ou **GML_EXPOSE_GML_ID** à **YES** ou **NO**.

À partir de OGR 1.9.0, lors de la création d'un document GML2, si un champ est appelé *fid*, son contenu sera également utilisé pour écrire le contenu de l'attribut fid de la feature créée.

F Problèmes lors de création

Lors de l'export, toutes les couches sont écrites dans un seul fichier GML dans une seule collection d'objet. Chaque nom de couche est utilisé comme nom d'élément pour les

objets à partir de cette couche. Les géométries sont toujours écrites comme un élément *ogr:geometryProperty* dans l'objet.

Le pilote GML gère en écriture les options de création de jeu de données suivantes :

- **XSISCHEMAURI :** si fournit, l'uri sera inséré comme location du schéma. Notez que le fichier schéma n'est pas réellement accédé par OGR, il est du ressort de l'utilisateur de s'assurer que le schéma correspond au fichier données GML produit par GML.
- **XSISCHEMA :** peut être *EXTERNAL*, *INTERNAL* ou *OFF* et par défaut à *EXTERNAL*. Cela écrit un fichier schéma GML vers un fichier .xsd correspondant (avec le même nom). Si *INTERNAL* est utilisé le schéma est écrit dans le fichier GML, mais cela est expérimental et probablement pas valide XML. *OFF* désactive la génération du schéma (et est implicite si *XSISCHEMAURI* est utilisé).
- **FORMAT :** (OGR >= 1.8.0) peut être définie à GML3 pour écrire des fichiers GML qui suivent le profile GML3 SF-0. Autrement GML2 sera utilisé.
- **GML3_LONGSRS=YES/NO :** (OGR >= 1.8.0, seulement valide quand FORMAT=GML3) YES par défaut. Si YES, SRS avec l'autorité EPSG sera écrit avec le préfixe "urn:ogc:def:crs:EPSG::". Dans ce cas, si la projection est une projection géographique sans ordre d'axe explicite, mais avec ce même code d'autorité de la projection importé avec *ImportFromEPSGA()* doit être traité comme lat/long, alors la fonction prendra soin d'échanger l'ordre des coordonnées. Si définie à NO, la projection avec l'autorité EPSG sera écrit avec le préfixe "EPSG:", même s'ils sont dans l'ordre lat/long.
- **SPACE_INDENTATION=YES/NO :** (OGR >= 1.8.0) YES par défaut. Si YES, la sortie sera indentée avec des espaces pour une meilleure lisibilité, mais avec une augmentation de la taille.

G Syntax of .gfs file by example

Considérons le fichier test.gml suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml">
    <gml:featureMember>
        <LAYER>
            <attrib1>attrib1_value</attrib1>
            <attrib2container>
                <attrib2>attrib2_value</attrib2>
            </attrib2container>
            <location1container>
                <location1>
                    <gml:Point><gml:coordinates>3,50</gml:coordinates>
</gml:Point>
                </location1>
            </location1container>
            <location2>
                <gml:Point><gml:coordinates>2,49</gml:coordinates></gm
l:Point>
            </location2>
        </LAYER>
    </gml:featureMember>
</gml:FeatureCollection>
```

et le fichier associé .gfs suivant :

```
<GMLFeatureClassList>
    <GMLFeatureClass>
        <Name>LAYER</Name>
        <ElementPath>LAYER</ElementPath>
        <GeometryElementPath>location1container|
location1</GeometryElementPath>
        <PropertyDefn>
            <Name>attrib1</Name>
            <ElementPath>attrib1</ElementPath>
            <Type>String</Type>
            <Width>13</Width>
        </PropertyDefn>
        <PropertyDefn>
            <Name>attrib2</Name>
            <ElementPath>attrib2container|attrib2</ElementPath>
            <Type>String</Type>
            <Width>13</Width>
        </PropertyDefn>
    </GMLFeatureClass>
</GMLFeatureClassList>
```

Notez la présence du caractère '|' dans les éléments <ElementPath> et <GeometryElementPath> pour définir l'élément géométrie/champ désiré qui est l'élément XML imbriqué. Les éléments champs imbriqués sont seulement géré à partir d'OGR 1.8.0, ainsi que spécifier <GeometryElementPath>. Si GeometryElementPath n'est pas définie, le pilote GML utilisera le dernier élément géométrie reconnu.

La sortie de *ogrinfo test.gml -ro -al* est :

```
Layer name: LAYER
Geometry: Unknown (any)
Feature Count: 1
Extent: (3.000000, 50.000000) - (3.000000, 50.000000)
Layer SRS WKT:
(unknown)
Geometry Column = location1container|location1
attrib1: String (13.0)
attrib2: String (13.0)
OGRFeature(LAYER):0
    attrib1 (String) = attrib1_value
    attrib2 (String) = attrib2_value
    POINT (3 50)
```

H Exemple

La commande ogr2ogr peut être utilisé pour faire un dump des résultats d'une requête Oracle en GML :

```
ogr2ogr -f GML output.gml OCI:usr/pwd@db my_feature -where "id = 0"
```

La commande ogr2ogr peut être utilisé pour faire un dump des résultats d'une requête PostGIS en GML :

```
ogr2ogr -f GML output.gml PG:'host=myserver dbname=warmerda' -sql
    "SELECT pop_1994 from canada where province_name = 'Alberta'"
```

I Voir aussi

- Spécifications du GML
- Profile GML 3.1.1 simple features
- Xerces
- :ref:`gdal.ogr.format.nas`
  Unknown interpreted text role "ref".

Chapitre CIV GMT ASCII Vectors (.gmt)

OGR gère la lecture et l'écriture au format vecteur ASCII de GMT. C'est le format utilisé par le paquet *Generic Mapping Tools* (GMT), et inclus les ajouts récents au format pour prendre en charge les types de géométries, et les attributs. Pour l'instant les fichiers GMT sont seulement géré s'ils ont l'extension ".gmt". Les anciens (simples) fichiers GMT sont traité soit comme des fichiers points soit des fichiers lignes selon si une ligne de ">" est rencontré avant le premier vertex. Les nouveau styles de fichiers ont une variété d'information auxiliaire incluant le type de la géométrie, les étendues des couches, le système de coordonnées et les déclarations des champs attributaires en commentaire dans l'en-tête, et pour chaque objet géométrique peut avoir des attributs.

A Problème de création

Le pilote gère la création de nouveau fichiers GMLT et l'ajout d'objet géométriques additionnel à des fichiers existant, mais la mise à jour d'objet géométrique existant n'est pas géré. Chaque couche est créer comme un fichier .gmt séparé.

Chapitre CV GPSBabel

(GDAL/OGR >= 1.8.0)

Le pilote GPSBabel repose sur la commande GPSBabel pour accéder aux différents formats de fichier GPS.

L'exécutable GPSBabel doit être accessible à travers le PATH.

A Gestion de la lecture

Le pilote nécessite le pilote :ref:`gdal.ogr.formats.gpx` pour gérer entièrement la lecture (via la bibliothèque Expat) et être capable de parser la sortie de GPSBabel, puisque GPX est utilisé comme format intermédiaire.

Unknown interpreted text role "ref".

Les couches renvoyées peuvent être des waypoints, routes, route_points, tracks, track_points en fonction des données en entrée.

La synthaxe pour définir une source de données en entrée est : *GPSBabel:gpsbabel_file_format[,gpsbabel_format_option]*:[features=[waypoints,][tracks,][routes]:]filename* où :

- *gpsbabel_file_format* est un des formats de fichier pris en charge par GPSBabel.
- *gpsbabel_format_option* est n'importe quelles options pris en charge par le format spécifié par GPSBabel (référez vous au document de chaque format de GPSBabel)
- *features*= peut être utilisé pour modifier le type de feature que GPSBabel importera. waypoints correspond à l'option -w de la ligne de commande de gpsbabel, tracks correspond à -t et routes correspond à -r. Cette option peut être utilisé pour nécessité un import de donnée complet à partir du recepteur GPS qui sont lent et pour lesquels GPSBabel pourrait ne récupérer que les waypoints par défaut. Voyez la documentation sur Route and Track modes pour plus de détails.
- *filename* peut être un fichier réel sur disque, un fichier via l'API des fichiers virtuels de GDAL, ou un matériel spécial pris en charge par GPSBabel tel que "usb:", "/dev/ttyS0", "COM1:", etc.. Ce qui est réellement géré dépend du format de GPSBabel utilisé.

Alternativement, pour quelques formats de GPSBabel sélectionné, spécifier simplement le nom du fichier peut être suffisant. La liste pour l'instant est :

- garmin_txt
- gdb
- magellan
- mapsend
- mapsource
- nmea
- osm
- ozi

L'option de configuration *USE_TEMPFILE=YES* peut être utilisé pour créer un fichier GPX temporaire sur disque au lieu d'un en mémoire, lors de la lecture d'un gros volume de données.

B Gestion de l'écriture

Le pilote repose sur le pilote GPX pour créer un fichier intermédiaire qui sera finalement traduit par GPSBabel vers le format de GPSBabel désiré (le pilote GPX ne nécessite pas d'être configuré pour la gestion de la lecture pour la gestion de l'écriture de GPSBabel).

Les géométries gérées, options et autres problèmes de création sont ceux du pilote GPX. Référez vous à la page :ref:`gdal.ogr.formats.gpx` pour plus de détails.

Unknown interpreted text role "ref".

La syntaxe pour définir une source de données en sortie est : *GPSBabel:gpsbabel_file_format[,gpsbabel_format_option]*:filename* où :

- *gpsbabel_file_format* est un des formats de fichier pris en charge par GPSBabel.
- *gpsbabel_format_option* est une des options pris en charge par le format de GPSBabel définie (référez vous à la documentation des formats de GPSBabel)

Alternativement, vous pouvez juste passer un nom de fichier comme nom de source de données en sortie et définir l'option de création du jeu de données GPSBABEL_DRIVER=gpsbabel_file_format[,gpsbabel_format_option]*

L'option de configuration *USE_TEMPFILE=YES* peut être utilisé pour créer un fichier GPX temporaire sur disque au lieu d'un en mémoire, lors de l'écriture d'un gros volume de données.

1 Exemples

- Lire les waypoints à partir d'un récepteur USB Garmin :

```
ogrinfo -ro -al GPSBabel:garmin:usb:
```

- Convertir un shapefile vers un format Magellan Mapsend :

```
ogr2ogr -f GPSBabel GPSBabel:mapsend:out.mapsend in.shp
```

2 Voir également

- Home page de GPSBabel
- Formats de fichier de GPSBabel
- :ref:`gdal.ogr.formats.gpx`
  Unknown interpreted text role "ref".

Chapitre CVI GPX - GPS Exchange Format

(à partir de GDAL 1.5.0)

Le format GPX (format d'échange de GPS) est un format de données en XML léger pour l'échange de données GPS (waypoints, routes et tracks) entre des applications et des webservices sur Internet.

OGR gère la lecture et l'écriture du GPX (si GDAL est compilé avec la gestion de la bibliothèque *expat*).

Les versions gérées sont GPX 1.0 et 1.1 en lecture, GPX 1.1 en écriture.

Le pilote OGR gère la lecture et l'écriture de tous les types de géométrie GPX :

- **waypoints :** couche d'objet de type *wbbpoint* d'OGR ;
- **routes :** couche d'objet de type *wkbLineString* d'OGR ;
- **tracks :** couche d'objet de type *wkbMultiLineString* d'OGR.

Il gère également la lecture des points routes et des points tracks dans des couches indépendantes (route_points et track_points), ainsi leurs propres attributs peuvent être utilisé par OGR.

En plus de ses attributs GPX, chaque point route d'une route a un *route_fid* (clé étrangère au FID de sa route dont il fait partie) et un *route_point_id* qui est le numéro de séquence dans la route.

La même chose s'applique pour les points tracks avec *track_fid*, *track_seg_id* et *track_seg_point_id*. Toutes les coordonnées sont relatives au datum WGS84 (EPSG:4326).

Si la variable d'environnement *GPX_ELE_AS_25D* est définie à *YES*, l'élément d'élévation sera utilisé pour définir la coordonnée Z des waypoints, des points routes et des points tracks.

OGR/GPX lit et écrite les attributs GPX pour les waypoints, routes et tracks.

Par défaut, jusqu'à 2 éléments <link> peuvent être pris en compte par objet géométrique. Le nombre par défaut peut être changé par la variable d'environnement *GPX_N_MAX_LINKS*.

A Problèmes d'encodage

La bibliothèque Expat gère la lecture des encodages internes suivants :

- US-ASCII
- UTF-8
- UTF-16
- ISO-8859-1

OGR 1.8.0 ajoute la gestion des encodages Windows-1252 (pour les versions antérieures, modifier l'encodage mentionné dans l'en-tête XML à ISO-8859-1 peut fonctionner dans certains cas).

Le contenu retourné par OGR sera encodé en UTF-8, après la conversion à partir de l'encodage mentionné dans l'en-tête du fichier.

Si votre fichier GPX n'est pas encodé dans un de ces encodages précédents, il ne sera pas parsé par le pilote GPX. Vous devrez le convertir dans un des encodages gérés avec la commande ''iconv'' par exemple et changer en fonction de la valeur du paramètre d'encodage dans l'en-tête du XML.

Lors de l'écriture d'un fichier GPX, le pilote s'attend à du contenu en UTF-8.

B Lecture des l'élément extensions

Si l'élément <extensions> est détecté dans le fichier GPX, OGR exposera le contenu de ses sous-éléments comme champs. Un contenu complexe des sous-éléments sera exposé comme un blob XML.

Le contenu GPX de la séquence suivante :

```
<extensions>
    <navaid:name>TOTAL RF</navaid:name>
    <navaid:address>BENSALEM</navaid:address>
    <navaid:state>PA</navaid:state>
    <navaid:country>US</navaid:country>
    <navaid:frequencies>
    <navaid:frequency type="CTAF" frequency="122.900" name="CTAF"/>
    </navaid:frequencies>
    <navaid:runways>
    <navaid:runway designation="H1" length="80" width="80"
surface="ASPH-G">
    </navaid:runway>
    </navaid:runways>
    <navaid:magvar>12</navaid:magvar>
</extensions>
```

sera interprété dans un modèle *Simple Feature* d'OGR comme :

```
navaid_name (String) = TOTAL RF
navaid_address (String) = BENSALEM
navaid_state (String) = PA
navaid_country (String) = US
navaid_frequencies (String) = <navaid:frequency type="CTAF"
frequency="122.900" name="CTAF" ></navaid:frequency>
navaid_runways (String) = <navaid:runway designation="H1" length="80"
width="80" surface="ASPH-G" ></navaid:runway>
```

```
navaid_magvar (Integer) = 12
```

Note!

Le pilote GPX affichera le contenu de l'élément extension seulement s'il est trouvé dans les premiers enregistrements du fichier GPX. Si les extensions apparaissent plus tard, vous pouvez forcer le parsage explicite de l'ensemble du fichier avec la variable d'environnement GPX_USE_EXTENSIONS.

C Problème de création

Lors de l'export, toutes les couches sont écrites dans un seul fichier GPX. La mise à jour de fichier existant n'est pas pour l'instant gérée.

Si le fichier en sortie existe déjà, l'écriture ne se fera pas. Vous devez effacer le fichier existant d'abord.

Géométries gérées :

- les objets géométriques de type *wkbPoint/wkbPoint25D* sont écrits dans l'élément wpt.
- les objets géométriques de type *wkbLineString/wkbLineString25D* sont écrits dans l'élément rte.
- les objets géométriques de type *wkbMultiLineString/wkbMultiLineString25D* sont écrit dans l'élément trk.
- les autres types de géométrie ne sont pas gérés.

Pour les points des routes et les points de trace, s'il y a une coordonnée Z, celui-ci est utilisé pour remplir l'élément élévation des points correspondants.

À partir de GDAL/OGR 1.8.0, si une couche est nommée "track_points" avec des géométries wkbPoint/wkbPoint25D, le tracks dans le fichier GPX sera construit à partir de la séquence de features dans cette couche. C'est la manière de définir les attributs GPX pour chaque point d'une trace, en plus des coordonnées brutes. Les points appartenant au même trace sont identifiés grâce à la même valeur du champ 'track_fid' field (et sera découpé en segments de trace en fonction de la valeur du champ 'track_seg_id'). Ils doivent être écrits en séquence afin que les objets trace soient correctement reconstruit. Le champ 'track_name' peut être définie sur le premier point de la trace pour remplir l'élément <name> de la trace.

De la même manière, si une couche est nommée "route_points" avec des géométries wkbPoint/wkbPoint25D, les routes dans le fichiers GPX sera construit à partir de la séquence de points avec la même valeur du champ 'route_fid'. Le champ 'route_name' peut être définie au premier point trace pour remplir l'élément <name> sur la route.

Le pilote GPX gère en écriture les options de création suivantes pour les couches :

- **FORCE_GPX_TRACK :** par défaut lors de l'écriture d'une couche dont les objets géométriques sont de type *wkbLineString*, le pilote GPX choisit de les écrire comme routes.
  Si *FORCE_GPX_TRACK=YES* est définie, ils seront écrits comme tracks.
- **FORCE_GPX_ROUTE :** par défaut lors de l'écriture d'une couche dont les objets sont de type *wkbMultiLineString*, le pilote GPX choisit de les écrire comme tracks.
  Si *FORCE_GPX_ROUTE=YES* est définie, ils seront écrits comme routes,

seulement si les multilignes ne sont composées que de ligne simple.

Le pilote GPX gère en écriture les options de création suivantes pour les jeux de données :

- **GPX_USE_EXTENSIONS :** Par défaut, le pilote GPX ignorera les champs attributaires qui ne correspondront pas à la définition du schéma XML du GPX (name, cmt, etc...). Si *GPX_USE_EXTENSIONS=YES* est définie, des champs supplémentaires seront écrits dans la balise <extension>.
- **GPX_EXTENSIONS_NS :** Seulement utilisé si *GPX_USE_EXTENSIONS=YES* et *GPX_EXTENSIONS_NS_URL* sont définies. La valeur de l'espace de nom utilisée pour les balises extension. Par défaut, "ogr".
- **GPX_EXTENSIONS_NS_URL :** Seulement utilisé si *GPX_USE_EXTENSIONS=YES* et *GPX_EXTENSIONS_NS* sont définies. Le chemin de l'espace de nom est par défaut "http://osgeo.org/gdal".
- **LINEFORMAT :** (GDAL/OGR >= 1.8.0) Par défaut les fichiers sont créés avec la convetion de fin de ligne de la plateforme locale (CR/LF sur win32 ou LF sur tous les autres systèmes). Cela peut être écrasé par l'utilisation de l'option de création de couche LINEFORMAT qui accepte les valeurs **CRLF** (format DOS) ou **LF** (format Unix).

Waypoints, routes et tracks doivent être écrit dans cet ordre en fonction du schéma XML.

Lors de la traduction à partir d'une source de jeu de données, il peut être nécessaire de renommer les noms des champs à partir de la source du jeu de données par les noms des attributs GPX attendus, tels que <name>, <desc>, etc. Cela peut être réalisé avec un jeu de données :ref:`gdal.ogr.formats.vrt`, ou en utilisant l'option "-sql" de la commande `ogr2ogr`.

Unknown interpreted text role "ref".

D Problèmes lors de la traduction en Shapefile

- Lors de la traduction de la couche *track_points* vers un Shapefile, les noms des champs "track_seg_id" et "track_seg_point_id" sont tronqué en 10 caractères dans le fichier .DBF, entraînant une duplication du nom. Pour éviter cela à partir de GDAL 1.6.1, vous pouvez définir l'option de configuration GPX_SHORT_NAMES à TRUE pour les définir respectivement à "trksegid" et "trksegptid", ce qui leur permet d'être unique une fois traduit en DBF. Le champ "route_point_id" de la couche *route_points* sera également renommé en "rteptid". Mais notez qu'aucun traitement particulier ne sera réalisé pour n'importe quelle extension de noms de champ.
  Pour traduire la couche track_points d'un fichier GPX à un ensemble de shapefiles :

```
ogr2ogr --config GPX_SHORT_NAMES YES out input.gpx track_points
```

- Shapefile ne gère pas les champs de type DateTime. Il gère seulement les champs de type Date. Par défaut, vous perdrez donc la partie hour:minute:second de l'élément *Time* d'un fichier GPX. À partir de GDAL 1.6.0, vous pouvez utiliser l'opérateur CAST du SQL d'OGR pour convertir le champ *time* en string :

```
ogr2ogr out input.gpx -sql "SELECT ele, CAST(time AS
character(32)) FROM waypoints"
```

  À partir de GDAL 1.7.0, il y a des façons plus aisées pour sélectionner tous les

champs et demander la conversion d'un type donné en strings :

```
ogr2ogr out input.gpx -fieldTypeToString DateTime
```

E Exemple

La commande ogrinfo peut être utilisée pour faire un dump du contenu d'un fichier de données GPX :

```
ogrinfo -ro -al input.gpx
```

La commande ogr2ogr peut être utilisé pour une traduction du format GPX au format GPX :

```
ogr2ogr -f GPX output.gpx input.gpx waypoints routes tracks
```

Note!

Dans le cas de la traduction du format GPX au format GPX, vous devez définir le nom des couches, dans le but d'éviter les couches route_points et track_points.

Utilisez la baliser <extensions> pour la sortie :

```
ogr2ogr -f GPX  -dsco GPX_USE_EXTENSIONS=YES output.gpx input
```

qui renverra ce qui suit :

```
<?xml version="1.0"?>
<gpx version="1.1" creator="GDAL 1.5dev"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ogr="http://osgeo.org/gdal"
xmlns="http://www.topografix.com/GPX/1/1"
xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd">
<wpt lat="1" lon="2">
<extensions>
    <ogr:Primary_ID>PID5</ogr:Primary_ID>
    <ogr:Secondary_ID>SID5</ogr:Secondary_ID>
</extensions>
</wpt>
<wpt lat="3" lon="4">
<extensions>
    <ogr:Primary_ID>PID4</ogr:Primary_ID>
    <ogr:Secondary_ID>SID4</ogr:Secondary_ID>
</extensions>
</wpt>
</gpx>
```

Utilisez l'option -sql pour remaper les noms des champs par un permis par le schéma GPX !

```
ogr2ogr -f GPX output.gpx input.shp -sql "SELECT field1 AS name,
```

```
field2 AS desc FROM source_layer"
```

F Voir également

- Page web pour le format GPX format
- Documenation du format GPX 1.1

Chapitre CVII GRASS

Le pilote GRASS peut lire les cartes vectorielles GRASS (version 6.0 et au delà). Chaque carte vectorielle GRASS est représentée comme une source de données. Une carte vectorielle de GRASS peut avoir 0, 1 ou plusieurs couches.

Les points de GRASS sont représenté comme *wkbPoint*, les lignes et limites comme *wkbLineString* et les surfaces comme *wkbPolygon*. *wkbMulti** et *wkbGeometryCollection* nesont utilisé. Des objets géométriques peuvent être mélangé dans une couche. Si une couche contient seulement des objets d'un seul type, il est définie par le type approprié et peut être récupéré par *OGRLayer::GetLayerDefn();*

Si une géométrie a plus de catégories de la même couche lié, il est représenté en autant d'objet (une pour chaque catégorie).

À la fois les cartes 2D et 3D sont gérées.

A Nom de la source de données

Le nom de la source de données est le chemin complet vers le fichier 'head' dans le répertoire vector/ de GRASS. En utilisant les noms des variables d'environnement de GRASS, cela peut être exprimé par :

```
$GISDBASE/$LOCATION_NAME/$MAPSET/vector/mymap/head
```

où 'mymap' est le nom d ela carte vectorielle. Par exemple :

```
/home/cimrman/grass_data/jizerky/jara/vector/liptakov/head
```

B Noms de la couche

Habituellement les numéros de la couche sont utilisé comme noms de couche. Le numéro 0 d'une couche est utilisé pour tous les objets sans catégorie. Il est possible de donner en option le nom de la couche GRASS lié à la base de données cependant ce n'est pas encore géré par les modules GRASS. Un nom de couche peut être ajouté dans le fichier vecteur 'dln' comme '/name' après le numéro de couche, par exemple à l'enregistrement

original :

```
1 rivers cat $GISDBASE/$LOCATION_NAME/$MAPSET/dbf/ dbf
```

il est possible d'assigner le nom 'rivers'

```
1/rivers rivers cat $GISDBASE/$LOCATION_NAME/$MAPSET/dbf/ dbf
```

la couche 1 qui sera listée est la couche 'rivers'.

C Filtre attributaire

Si une couche a des attributs stockés dans une base de données, la requête est passée au pilote de la base de données sous-jacente. Ce qui signifie que les conditions SQL qui peuvent être utilisées dépendent des pilotes et de la base de données à laquelle la couche est liée. Par exemple, le pilote DBF a pour l'instant que très peu d'expressions SQL et PostgreSQL offrent un ensemble d'expression très riche.

Si une couche n'a pas d'attributs liés et n'a que des catégories, le moteur SQL interne à OGR est utilisé pour évaluer l'expression. Les catégories sont des nombres entiers attaché à une géométrie, c'est une sorte d'identifiant, mais ce n'est pas un FID puisque plusieurs objets géométriques dans une couche peuvent avoir la même catégorie.

L'évaluation est réalisée lorsque le filtre des attributs est défini.

D Filtre spatial

Les boites englobantes des objets géométriques stockées dans une structure topologique sont utilisées pour évaluer si des géométries correspondent au filtre spatial en cours.

L'évaluation est réalisée une fois que le filtre spatial est défini.

E GISBASE

GISBASE est le chemin complet vers le répertoire où GRASS est installé. Par défaut, le pilote GRASS utilise le chemin donné par le script de configuration de GDAL. Un répertoire différent peut être forcé en définissant la variable d'environnement *GISBASE*. *GISBASE* est utilisé pour trouver les pilotes des bases de données GRASS.

F Topologie manquante

Le pilote GRASS peut lire les fichiers vecteurs de GRASS si la topologie est disponible (aka niveau 2). Si une erreur est retournée, disant que la topologie n'est pas disponible, il est nécessaire de construire la topologie avec le module `v.build` de GRASS.

G Accès aléatoire

Si un accès aléatoire (*GetFeature* au lieu de *GetNextFeature*) est utilisé sur une couche avec des attributs, la lecture des géométries peut être assez lente. Cela est dû au fait que le pilote doit faire une requête sur les attributs par catégorie pour chaque objet géométrique (pour éviter d'utiliser beaucoup de mémoire) et l'accès aléatoire à une base de données est habituellement lent. Cela peut être amélioré du côté de GRASS lors de

l'optimisation et de l'écriture des fichiers basés sur des pilotes (DBF, SQLite).

H Problèmes connus

À cause d'un bug dans la bibliothèque de GRASS, il est impossible de démarrer/arrêter les pilotes des bases de données dans l'ordre FIFO et l'ordre FILO doit être utilisé. Le pilote GRASS pour OGR est écrit avec cette limitation à l'esprit et les pilotes sont toujours fermés s'ils ne sont pas utilisés et si nu pilote reste ouvert, la commande *kill()* est utilisée pour l'arrêter. Il peut arriver cependant dans certains cas que le pilote tente d'arrêter un pilote de base de données qui n'est pas le dernier ouvert et l'application se termine. Cela peut arriver si la lecture séquentielle (*GetNextFeature*) d'une couche n'est pas terminée (la lecture est arrêtée avant que la dernière géométrie disponible ne soit atteinte), les objets d'une autre couche sont lu puis la lecture de la première couche est terminée, parce que dans ce cas-là la commande *kill()* n'est pas utilisée.

I Voir également

- Page principale de GRASS

Le développement de ce pilote a été financé par Faunalia (http://www.faunalia.it).

Chapitre CVIII GTM - GPS TrackMaker

(à partir de GDAL 1.7.0)

GPSTrackMaker est nu programme qui est compatible avec plus de 160 modèles GPS. Il vous permet de créer vos propres cartes. Il gère les cartes vecteurs et images.

Le pilote OGR gère la lecture et l'écriture des fichiers GTM 211 (.gtm) ; cependant, dans cette implémentation nous ne gérons pas les images et les routes. Waypoints et traces sont gérés.

Bien que GTM gère plusieurs types de données, comme NAD 1967, SAD 1969, et d'autres, le fichier en sortie du pilote d'OGR utilisera WGS 1984. Le pilote GTM lira proprement seulement les fichiers GTM géoréférencés en WGS 1984 (si ce n'est pas le cas une alerte sera envoyée).

Le pilote OGR gère seulement les POINT, LINESTRING, et MULTILINESTRING.

A Exemple

- La commande `ogrinfo` peut être utilisée opur dumper le contenu d'un fichier de données GTM :

```
ogrinfo -ro -al input.gtm
```

- Utilisez l'option *-sql* pour mapper les noms des champs vers ceux autorisées par le schéma GTM :

```
ogr2ogr -f "GPSTrackMaker" output.gtm input.shp -sql "SELECT
field1 AS name, field2 AS color, field3 AS type FROM input"
```

- Exemple de traduction à partir de PostGIS vers GTM :

```
ogr2ogr -f "GPSTrackMaker" output.gtm PG:"host=hostaddress
user=username dbname=db password=mypassword" -sql "select filed1
as name, field2 as color, field3 as type, wkb_geometry from
input" -nlt MULTILINESTRING
```

Note!

vous devez définir le type de couche en tant que POINT, LINESTRING, ou MULTILINESTRING.

B Voir également

- Home page pour le Programme GPS TrackMaker
- Documentation du format GTM 211

Chapitre CIX HTF - Hydrographic Transfer Format

(GDAL/OGR >= 1.8.0)

Ce pilote lit les fichiers contenant des sondages qui suivent le Format de Transfert Hydrographique (HTF), qui est utilisé par l'*Australian Hydrographic Office* (AHO).

Le pilote a été développé en se basant sur la spécification 2.02 HTF.

Le fichier doit être géoréférencé en UTM WGS84 pour être considéré valide par le pilote.

Le pilote renvoie deux couches spatiales : une couche nommée "polygon" et une couche nommée "sounding".

Il y a également une couche "cachée", nommée "metadata", qui peut être récupéré par *GetLayerByName()*, et qui contient une seule feature, composée de lignes d'en-tête du fichier.

Polygons sont utilisés pour distinguer les différentes catégories de surveillance, tels que n'importe quel changement significatif dans la précision de la position/profondeur ou/et un changement dans la couverture des fonds marins entraînera un contour de polygone séparé contenant des polygones.

La couche "polygon" contient les champs suivants :

- *DESCRIPTION* : définie les polygones de chaque région avec des critères de surveillance ou de thème similaire.
- *IDENTIFIER* : identifiant unique du polygone pour cette transmission.
- *SEAFLOOR_COVERAGE* : toutes features de fond sous-marin significatif détecté (ensonification complète / balayer) ou couverture complète non achevée et des features non cartographiées peuvent exister.
- *POSITION_ACCURACY* : +/- NNN.n mètres à 95% CI (2.45) en fonction du datum donné.
- *DEPTH_ACCURACY* : +/- NN.n mètres à 95% CI (2.00) à des profondeurs critiques.

La couche "sondage" doit contenir - au minimum - les 20 champs suivants :

- *REJECTED_SOUNDING* : si 0 le sondage est valide ou si 1 le sondage a été rejeté (flagged).
- *LINE_NAME* : nom/numéro de ligne de l'enquête comme identifiant unique au

sein de l'enquête.
- *FIX_NUMBER* : numéro séquentiel fixe du sondage, unique dans l'enquête.
- *UTC_DATE* : date UTC pour le sondage CCYYMMDD.
- *UTC_TIME* : temps UTC pour le sondage HHMMSS.ss.
- *LATITUDE* : latitude du sondage +/-NNN.nnnnnn (degrés d'arc, le sud est négative).
- *LONGITUDE* : longitude du sondage +/-NNN.nnnnnn (degrés d'arc, l'est est négative).
- *EASTING* : Grille de coordonnée de la position du sondage en mètres NNNNNNN.n.
- *NORTHING* : Grille de coordonnée de la position du sondage en mètres NNNNNNN.n.
- *DEPTH* : valeur sonore réduite en mètres avec des corrections appliquées comme indiqué dans les champs concernés, les sondages sont positifs et les hauteurs de séchage sont négatifs +/-NNNN.nn mètres.
- *POSITIONING_SENSOR* : indique le numéro du système de position rempli dans l'enregistrement de l'en-tête du HTF.
- *DEPTH_SENSOR* : indique le numéro du système de sondage de la profondeur rempli dans l'enregistrement de l'en-tête du HTF.
- *TPE_POSITION* : erreur Totale propagée de la composante horizontale de la sonde.
- *TPE_DEPTH* : erreur totale propagée de la composante verticale de la sonde.
- *NBA FLAG* : option *No Bottom at*, si 0 pas de profondeur NBA ou si 1 la profondeur est NBA, des eaux plus profondeur existe probablement.
- *TIDE* : Valeur de la correction de la marée appliquée +/- NN.nn mètres.
- *DEEP_WATER_CORRECTION* : valeur de la vitesse de sondage de l'eau profonde appliquée +/- NN.nn mètres.
- *VERTICAL BIAS_CORRECTION* : valeur du biais vertical appliqué +/- NN.nn mètres. eg correction de la profondeur du transducteur.
- *SOUND_VELOCITY* : vitesse du son mesurée utilisé pour traiter le sondage en mètres par seconde IIII
- *PLOTTED_SOUNDING* : si 0 alors la profondeur réduite ne figurait pas sur le fairsheet original ou si 1 alors la profondeur réduite est apparu sur le fairsheet original.

Certains champs peuvent n'être jamais définie, en fonction de la valeur du champ de la Clé de Population du Champ. Les champs supplémentaires peuvent aussi être ajoutés.

A Voir également

- Page principale du format HTF - Hydrographic Transfer Format
- Spécification technique HTF

Chapitre CX IDB

Ce pilote implémente la gestion de l'accès aux tables spatiales dans un Informix d'IBM étendue avec le module spatial DataBlade.

Lors de l'ouverture d'une base de données, son nom doit être définie sous la forme :

```
"IDB:dbname={dbname} server={host} user={login} pass={pass}
table={layertable}".
```

Le préfixe *IDB:* est utilisé pour marquer le nom comme une chaine de connection IDB.

Si la table *geometry_columns* existe, alors toutes les tables listées et les vues nommées seront traitées comme des couches OGR. Autrement toutes les tables attributaires seront traitées comme des couches.

Les tables attributaires (non spatiale) peuvent être accédées, et renverront des objets avec des attributs mais sans géométrie. Si la table a un champ "*st_**", celui-ci sera traité comme un table spatiale. Le type du champ est analysé pour déterminer comment le lire.

Le pilote gère la détection automatique des FID.

A Variables d'environnement

- **INFORMIXDIR :** il doit être définie au répertoire d'installation du SDK client d'Informix
- **INFORMIXSERVER :** nom du serveur Informix par défaut
- **DB_LOCALE :** locale de la base de données d'Informix
- **CLIENT_LOCALE :** locale du client
- **IDB_OGR_FID :** définie le nom de la clé primaire au lieu de '*ogc_fid*'.

Pour plus d'information sur les variables d'Informix lisez la documentation du SDK client d'Informix.

B Exemple

Cet exemple montre l'utilisation de ''ogrinfo'' pour lister les couches DataBlade d'Informix sur un hôte différent :

```
ogrinfo -ro IDB:'server=demo_on user=informix dbname=frames'
```

Chapitre CXI INTERLIS

OGR gère le format INTERLIS en lecture et écriture.

INTERLIS est un standard qui a été spécialement créé dans le but de répondre aux besoins de modélisation et l'intégration de données spatiales dans des systèmes d'information géographique future et contemporain. La version actuelle est INTERLIS version 2. INTERLIS version 1 reste un standard suisse. Avec l'utilisation de données spatiales documenté et unifié et des possibilités d'échanges flexible les avantages suivantes apparaissent :

- la documentation standardisé ;
- l'échange de données compatible ;
- l'intégration complète des données spatiales par exemple à partir de différents fournisseur de données ;
- Test de qualité ;
- le stockage des données à long terme ;
- la sécurité du *contrat-preuve* et la disponibilité du logiciel.

A Modèle

Les données sont lues et écrites dans des fichiers de transfert qui ont des formats différent dans INTERLIS 1 (.itf) et INTERLIS 2 (.xml). Pour utiliser le modèle INTERLIS (.ili) un interpréteur Java est nécessaire à l'exécution, et le fichier ili2c.jar (inclus dans le compilateur pour INTERLIS 2 doit être dans le path. Le fichier modèle peut être ajouté au fichier de transfert séparé par une virgule.

Quelques transformations possible avec :ref:`gdal.ogr.ogr2ogr`.

Unknown interpreted text role "ref".

- Interlis 1 -> Shape :

```
ogr2ogr -f "ESRI Shapefile" shpdir ili-bsp.itf,ili-bsp.ili
```

- Interlis 2 -> Shape :

```
ogr2ogr -f "ESRI Shapefile" shpdir
RoadsExdm2ien.xml,RoadsExdm2ben.ili,RoadsExdm2ien.ili
```

ou sans model :

```
ogr2ogr -f "ESRI Shapefile" shpdir RoadsExdm2ien.xml
```

- Shape -> Interlis 1 :

```
ogr2ogr -f "Interlis 1" ili-bsp.itf
Bodenbedeckung__BoFlaechen_Form.shp
```

- Shape -> Interlis 2 :

```
ogr2ogr -f "Interlis 2" LandCover.xml,RoadsExdm2ben.ili
RoadsExdm2ben_10.Roads.LandCover.shp
```

- Import incrémental de Interlis 1 vers PostGIS :

```
ogr2ogr -f PostgreSQL PG:dbname=warmerda
av_fixpunkte_ohne_LFPNachfuehrung.itf,av.ili -lco OVERWRITE=yes
ogr2ogr -f PostgreSQL PG:dbname=warmerda
av_fixpunkte_mit_LFPNachfuehrung.itf,av.ili -append
```

B Interpolation d'arc

Les géométries d'arc d'INTERLIS sont converties en polygone.

L'angle d'interpolation peut être changé avec la variable d'environnement *ARC_DEGREES* (Par défaut : 1 degré).

C Autres remarques

- Plus d'information : http://gis.hsr.ch/wiki/OGR (en allemand)
- Le développement du pilote ONTERLIS d'OGR a été financé par Swiss Federal Administration, Canton Solothurn et Canton Thurgovia.

Chapitre CXII INGRES

Le pilote implémente la lecture et l'écriture pour les données spatiales dans des tables de base de donnés INGRES. Cette fonctionnalité a été introduite dans GDAL/OGR 1.6.0.

Lors de l'ouverture d'une base de données, son nom doit être définie dans la forme *@driver=ingres,dbname=dbname[,options]* où les options peuvent être listées séparé par des virgules comme *username*=**userid*, password=*password*, timeout=*timeout*, tables=table1/table2*. Le pilote (driver) et les valeurs de *dbname* sont nécessaires alors que le reste est optionnel. Si le nom de l'utilisateur et le mot de passe ne sont pas fournie une tentative est faite pour s'authentifier comme l'utilisateur encors.

Exemples :

```
@driver=ingres,dbname=test,userid=warmerda,password=test,tables=usa/ca
nada

@driver=ingres,dbname=mapping
```

Si la liste des tables n'est pas fournie, une tentative est réalisée pour énumérer toutes les tables non système comme couches, autrement seul les tables listées sont représentées comme couches. Cette option est surtout utile lorsqu'une base de données possède beaucoup de tables, et scanner tous les schémas pourrait prendre trop de temps.

Si un champ entier nommé "ogr_fid" existe dans une table celui-ci sera nommée comme FID, autrement les FIDs seront assignés séquentiellement. Cela implique que différents FIDs seront assigné à une enregistrement/objets en fonction des filtres de requête attributaire et spatial en un temps données.

Par défaut, les requêtes SQL sont envoyé directement au moteur de base de données INGRES. Il est aussi possible de réaliser une requête au pilote pour prendre en charge les commandes SQL avec le moteur SQL d'OGR en envoyant la chaîne "OGRSQL" à la méthode *ExecuteSQL()* comme nom du dialecte SQL. Pour l'instant le pilote INGRES gère seulement les vieux types de données spatiales d'INGRES tels que *POLYGON*, *POINT*, *LINE*, *LONG POLYGON*, *LONG LINE*, etc. Il est prévue dans le futur qu'un nouvel ensemble de type spatial conforme à l'OGC soit géré.

A Avertissements

- Les types spéciaux CIRCLE et ICIRCLE ne sont pas géré pour l'instant en lecture.
- Aucun index spatial rapide n'est utilisé lors de la lecture, les filtres spatiaux sont donc implémenté en lisant et parsant tous les enregistrements, et ceux qui ne satisferaient pas le filtre spatial sont ignorés.
- Il n'y a pas de gestion pour les systèmes de coordonnées.

B Problèmes lors de la création

Le pilote INGRES ne gère pas la création de nouveaux jeux de données (une base de données dans INGRES) mais il permet la création de nouvelles couches (tables) dans une instance de base de données existante.

- Le pilote INGRES ne permet pas l'encodage de caractères pour le moment.
- Le pilote INGRES n'est pas transactionnel pour le moment.
- les types spatiaux BOX et IBOX ne sont pas gérés pour la création et lorsqu'ils sont lu sont représentés comme des polygones.
- Les types *non-LONG* (tels que *LINE*, *ILINE*, et *POLYGON*) gère seulement un nombre limité de sommet. Toute tentative pour créer des objets avec plus que le maximum de sommet possible pour cette couche échouera.
- Les vieux types spatiaux d'ingres sont très particulier pour les validations géométriques et toute tentative pour insérer (créer) des objets avec des géométries invalides échouera. Les raisons de l'invalidité incluent auto-intersection de polyligne, et auto-intersection de polygones.

C Options de création de couche

- **OVERWRITE :** il peut être à "YES" pour forcer une couche existante au nom désiré à être détruite avant la création de la couche voulues.
- **LAUNDER :** il peut être à "YES" pour forcer les nouveaux champs créés sur cette couche pour avoir leur noms de champs "nettoyés" dans une forme compatible avec MySQL. Cela les passe les caractères en minuscule et convertie certains caractères spéciaux comme "-" et "#" en "_". Si "NO" les noms exactes sont préservés. La valeur par défaut est "YES".
- **PRECISION :** il peut être à "TRUE" pour tenter de préserver la largeur et la précisions des champs pour la création et la lecture des couches MySQL. La valeur par défaut est "TRUE".
- **GEOMETRY_NAME :** cette option définie le nom de la colonne géométrique. La valeur par défaut est "SHAPE".
- **INGRES_FID :** cette option définie le nom de la colonne FID. La valeur par défaut est "OGR_FID".
- **GEOMETRY_TYPE :** définie le type d'objet pour la colonne géométrique. Il peut être l'un de *POINT*, *LSEG*, *LINE*, *LONG LINE*, *POLYGON*, ou *LONG POLYGON*. Par défaut *POINT*, *LONG LINE* ou *LONG POLYGON* sont utilisés en fonction du type de couche.

Chapitre CXIII KML - Keyhole Markup Language

Keyhole Markup Language (KML) est un langage basé sur XML pour la gestion de l'affichage des données géospatiales 3D. KML a été accepté comme standard OGC, et est géré d'une manière ou d'une autre par les navigateurs spatiaux majeurs. Remarquez que KML par définition utilise seulement une seule projection, *EPSG:4326*. Tous les exports KML d'OGR se présenteront en *EPSG:4326*. Par conséquent, OGR créera toutes les couches dans le système de coordonnées correcte et transformera toutes les géométries.

Pour l'instant, seulement les couches vectorielles sont prises en charge par le pilote KML *(il existe des scripts supplémentaires fournies avec le projet GDAL qui peuvent construire des export d'autres sortes)*.

A Lecture du KML

La lecture du KML est seulement disponible si GDAL/OGR est compilé avec le parser XML Expat, autrement seule l'écriture du KML sera gérée.

Les types de géométrie gérés sont Point, Linestring, Polygon, MultiPoint, MultiLineString, MultiPolygon et MultiGeometry. Il y a des limitations également, par exemple : les répertoires inclus dans un fichier KML source sont perdus ; les balises <description> des répertoires ne sera pas transmit à la sortie. Depuis GDAL 1.6.1, les répertoires contenant des types de géométries multiples, comme les points et les polygones, sont gérés.

B Écriture du KML

Puisque toutes les géométries du KML ne sont pas représenté dans le modèle de géométrie *Features*, vous ne pourrez pas générer plusieurs attributs spécifiques au KML à partir de GDAL/OGR. Testez quelques fichiers pour voir ce qui est possible.

Lors de l'export du KML, le pilote KML d'OGR traduira chaque couche OGR dans un répertoire KML, vous pouvez rencontrer des comportements inattendus si vous tentez de mélanger les types de géométries d'éléments dans une couche, par exemple des données *LINESTRING* et *POINT*.

Le pilote KML renommera certaines couches, ou des noms de répertoires du KML

source, par des noms qu'il considère valide, par exemple 'Layer #0', le nom par défaut de la première couche non nommée, deviendra 'Layer__0'.

KML est un mélange de mise en forme et de données géométriques. La balise <description> d'un *Placemark* sera affiché dans la plupart des navigateurs spatiaux comme une bulle contenant du html. Lors de l'écriture du KML, les attributs de l'élément *Layer* sont ajoutés comme de simples champs schéma. Cela préserve au mieux l'information du type de l'objet.

Une gestion limitée est disponible pour le remplissage (fill), couleur des lignes et autres attributs de styles. Essayez sur des échantillons pour avoir une meilleure vision des possibilités.

1 Problèmes d'encodage

La bibliothèque Expat gère la lecture des encodages internes suivants :

- US-ASCII
- UTF-8
- UTF-16
- ISO-8859-1

OGR 1.8.0 ajoute la gestion des encodages pour Windows-1252 (pour les versions précédentes, modifier l'encodage mentionné dans l'en-tête XML en ISO-8859-1 peut fonctionner dans certains cas).

Le contenu retourné par OGR sera encodé en UTF-8, après la conversion de l'encodage mentionné dans l'en-tête du fichier.

Si votre fichier KML n'est pas encodé dans l'un des encodages précédents, il ne sera pas parsé par le pilote KML. Vous devrez le convertir dans l'un des encodages gérés par la commande *iconv* par exemple et changer la valeur du paramètre *encoding* dans l'en-tête XML.

Lors de l'écriture du fichier KML, le pilote s'attend à du contenu en UTF-8.

2 Options de création

Les options de création suivantes sont gérées :

- **NameField :** permet de définir le champ à utiliser pour l'élément <name> du KML. La valeur par défaut est 'Name'.

```
ogr2ogr -f KML output.kml input.shp -dsco NameField=RegionName
```

- **DescriptionField :** permet de définir le champ à utiliser pour l'élément <description> du KML. La valeur par défaut est 'Description'
- **AltitudeMode :** permet de définir *AltitudeMode* à utiliser pour les géométries du KML. Cela affectera seulement les géoémtries 3D et doit être une des options KML valide. Lisez la documentation de référence du KML pour plus de détails.

```
ogr2ogr -f KML output.kml input.shp -dsco AltitudeMode=absolute
```

C Exemple

La commande `ogr2ogr` peut être utilisé pur faire un dump d'une requête PostGIS vers le

format KML :

```
ogr2ogr -f KML output.kml PG:'host=myserver dbname=warmerda' -sql
"SELECT pop_1994 from canada where province_name = 'Alberta'"
```

Pour faire un dump d'un fichier .kml comme OGR le voit :

```
ogrinfo -ro somedisplay.kml
```

D Avertissement

Google Earth semble avoir quelques limites avec le nombre de coordonnées dans des géométries complexes comme les polygone. Si le problème apparait, les géométries problématiques sont affichées complètement ou partiellement couvert par des rayures verticales. malheureusement, il n'y a pas de nombre exact donné dans la spécification KML à propos de cette limitation, le pilote KML ne préviendra pas sur des problèmes potentiels. Un des solutions testées possibles est de simplifier les lignes ou les polygones pour enlever quelques coordonnées. Voici la discussion sur ce problème sur le Forum développeur de Google KML, dans le thread polygone affiché avec des rayures verticales.

E Voir également

- Spécification KML
- Cours KML

Chapitre CXIV Pilote LIBKML (.kml .kmz)

Le pilote LIBKML est un client de Libkml de Google, une implémentation de référence du KML en lecture et écriture, sous la forme d'une biliothèque C++ multi-plateforme. Vous devez compiler et installer Libkml dans le but d'utiliser ce pilote OGR.

Notez que si vous compilez et incluez le pilote LIBKML, il deviendra le lecteur par défaut du KML pour OGR, écrasant le pilote KML précédent. Vous pouvez toujours définir soit KML ou LIBKML comme pilote de sortie via la ligne de commande.

Libkml de Google fournit des services de lecture pour un fichier KML valide. Toutefois, s'il vous plaît notez que certaines installations KML ne correspondent pas aux spécifications Simple Feature qu'ogr utilise comme structure interne. Par conséquent, un meilleur effort sera fait par le pilote pour comprendre le contenu d'un dossier KML lu par libkml dans OGR, mais votre version peut varier. S'il vous plaît essayez quelques fichiers KML comme échantillons pour avoir une idée de ce qui est bien interprété. En particulier, l'imbrication de features définit dans plus d'une profondeur seront aplaties pour être géré par le format interne d'OGR.

A Datasource

Vous pouvez définir un datasource comme fichier kml `somefile.kml`, un répertoire `somedir/`, ou un fichier kmz `somefile.kmz`.

Par défaut, sur les sources de données kmz et répertoire, un fichier index de toutes les couches sera lu ou écrit à partir ou vers le fichier doc.xml. Il contient <NetworkLink> pour chaque fichier couche dans le datasource. Cette fonctionnalité peut être désactivée en définissant la variable d'environnement *LIBKML_USE_DOC.KML* à "no".

1 StyleTable

Les tables des styles des datasource sont écrit dans le <Document> dans un .kml, style/style.kml dans un fichier kmz, or style.kml dans un répertoire, sous forme d'un ou plusieurs éléments <Style>. Tous les styles de feature d'OGR ne peuvent être traduit en KML.

B Layer

Layers sont déclarées dans les fichiers kml en tant que <Document> ou <Folder>, et dans les fichiers kmz ou les répertoires comme des fichiers kml séparés.

C Style

Les tables de style des couches ne peuvent pas être lues ou écrites à partir de/vers une couche kml qui est un <Folder>, sinon ils sont dans le <Document> d'une couche.

D Schéma

La lecture et l'écriture de <Schema> est géré pour les fichiers .kml, .kmz et les répertoires.

E Feature

Une feature OGR se traduit en kml en tant que <Placemark>.

1 Style

Les chaînes de style au niveau de la feature sont déclarées dans le KML soit en tant que <Style> ou <StyleUrl> dans chaque <Placemark>.

Lors de la lecture d'une feature kml et que la variable d'environnement *LIBKML_RESOLVE_STYLE* est positionnée à yes, les urls de styles sont recherchés dans les tableaux de style et les chaînes de style des features sont défini sur le style à partir de la table. C'est pour permettre la lecture de styles partagés par les applications, comme MapServer, qui ne lisent pas de tables de style.

Lors de la lecture d'une feature kml et que la variable d'environnement *LIBKML_EXTERNAL_STYLE* est positionnée à yes, une url de style qui est externe à la source de données est lu à partir du disque ou récupérés sur le serveur et analysée dans la table de style de la datasource. Si le style KML ne peut être lu ou *LIBKML_EXTERNAL_STYLE* est positionnée à no, alors l'url du style est copiée à la chaîne de style.

F Champs

Les champs OGR (attributs des feature) sont traduit vers le kml avec <Schema> et <SimpleData>, sauf pour certains champs spéciaux comme noté ci-dessous.

Un ensemble riche de variables d'environnement est disponible pour définir comment les champs en entrée et en sortie, sont traduit en kml <Placemark>. Par exemple, si vous voulez un champ appelé 'Cities' pour être traduit dans la balise <name> en KML, vous pouvez définir une variable d'environnement.

- **Name :**

  Ce champ de caractère traduit vers le kml la balise <name>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_NAME_FIELD*.
- **description :**

Ce champ de caractère traduit vers le kml la balise <description>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_DESCRIPTION_FIELD*.

- **timestamp :**

  Ce champ de caractère ou datetime ou date et/ou time traduit vers le kml la balise <timestamp>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_TIMESTAMP_FIELD*.
- **begin :**

  Ce champ de caractère ou datetime ou date et/ou time traduit vers le kml la balise <begin>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_BEGIN_FIELD*.
- **end :**

  Ce champ de caractère ou datetime ou date et/ou time traduit vers le kml la balise <end>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_END_FIELD*.
- **altitudeMode :**

  Ce champ de caractère traduit vers le kml la balise <altitudeMode> ou <gx:altitudeMode>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_ALTITUDEMODE_FIELD*.
- **tessellate :**

  Ce champ d'entier traduit vers le kml la balise <tessellate> Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_TESSELLATE_FIELD*.
- **extrude :**

  Ce champ d'entier traduit vers le kml la balise <extrude> Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_EXTRUDE_FIELD*.
- **visibility :**

  Ce champ d'entier traduit vers le kml la balise <visibility>. Le nom du champ ogr peut être changé avec la variable d'environnement *LIBKML_VISIBILITY_FIELD*.
- **OGR_STYLE :**

  Ce champ de caractère traduit vers un style de feature, OGR li ce champs s'il n'y a pas de chaînes de style définie sur la feature.

G Géométrie

Traduction de la Geometry d'OGR vers la géométrie KML est assez simple, avec seulement quelques exceptions. Point vers <Point>, LineString vers <LineString>, LinearRing vers <LinearRing>, et Polygon vers <Polygon>.

Dans OGR un polygone contient un tableau de LinearRings, le premier est celui à l'extérieur. KML a la balise <outerBoundaryIs> et <innerBoundaryIs> pour différencier les deux. OGR possède plusieurs types Multi des géométries : GeometryCollection,

MultiPolygon, MultiPoint, et MultiLineString. Quand cela est possible, OGR tentera de traduire <MultiGeometry> vers le type de géométrie OGR la plus précise (MultiPoint, MultiLineString ou MultiPolygon), et pas défaut vers GeometryCollection dans le cas de contenu mixte.

Parfois, la géométrie kml couvrira la Dateline, dans des applications comme QGIS ou MapServer cela va créer des lignes horizontales sur tout le pourtour du globe. En réglant la variable d'environnement *LIBKML_WRAPDATELINE* à "yes", contraindra le pilote libkml à diviser la géométrie à la Dateline lorsqu'il est lu.

H Exemple

Le script bash suivant construira un fichier csv et un fichier :ref:`gdal.ogr.formats.vrt`, puis les traduira en KML en utilisant :ref:`gdal.ogr.ogr2ogr` dans un fichier .kml avec un timestamp et des styles.

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

```
#!/bin/bash
# Copyright (c) 2010, Brian Case
#
# Permission is hereby granted, free of charge, to any person
obtaining a
# copy of this software and associated documentation files (the
"Software"),
# to deal in the Software without restriction, including without
limitation
# the rights to use, copy, modify, merge, publish, distribute,
sublicense,
# and/or sell copies of the Software, and to permit persons to whom
the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
included
# in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.


icon="http://maps.google.com/mapfiles/kml/shapes/shaded_dot.png"
rgba33="#FF9900"
rgba70="#FFFF00"
rgba150="#00FF00"
```

```
rgba300="#0000FF"
rgba500="#9900FF"
rgba800="#FF0000"

function docsv {

    IFS=','

    while read Date Time Lat Lon Mag Dep
    do
        ts=$(echo $Date | sed 's:/:-:g')T${Time%%.*}Z
        rgba=""

        if [[ $rgba == "" ]] && [[ $Dep -lt 33 ]]
        then
            rgba=$rgba33
        fi

        if [[ $rgba == "" ]] && [[ $Dep -lt 70 ]]
        then
            rgba=$rgba70
        fi

        if [[ $rgba == "" ]] && [[ $Dep -lt 150 ]]
        then
            rgba=$rgba150
        fi

        if [[ $rgba == "" ]] && [[ $Dep -lt 300 ]]
        then
            rgba=$rgba300
        fi

        if [[ $rgba == "" ]] && [[ $Dep -lt 500 ]]
        then
            rgba=$rgba500
        fi

        if [[ $rgba == "" ]]
        then
            rgba=$rgba800
        fi



        style="\"SYMBOL(s:$Mag,id:\"\"$icon\"\",c:$rgba)\""

        echo $Date,$Time,$Lat,$Lon,$Mag,$Dep,$ts,"$style"
    done

}


wget http://neic.usgs.gov/neis/gis/qed.asc -O /dev/stdout |\
tail -n +2 > qed.asc

echo
```

```
Date,TimeUTC,Latitude,Longitude,Magnitude,Depth,timestamp,OGR_STYLE >
qed.csv

docsv < qed.asc >> qed.csv

cat > qed.vrt << EOF
<OGRVRTDataSource>
    <OGRVRTLayer name="qed">
        <SrcDataSource>qed.csv</SrcDataSource>
        <GeometryType>wkbPoint</GeometryType>
        <LayerSRS>WGS84</LayerSRS>
        <GeometryField encoding="PointFromColumns" x="Longitude"
y="Latitude"/>
    </OGRVRTLayer>
</OGRVRTDataSource>

EOF

ogr2ogr -f libkml qed.kml qed.vrt
```

Chapitre CXV Access MDB databases

GDAL/OGR >= 1.9.0

OGR gère en option l'accès en lecture les fichiers .mdb en utilisant la bibliothèque Java Jackcess.

Ce pilote a d'abord comme objectif d'être utilisé sur les plateformes Unix pour contourner les problèmes avec la bibliothèque MDBTools qui agit comme le pilote ODBC pour les bases de données MDB.

Le pilote peut détecter les bases de données MDB de Geomedia et les géodatabases personnels d'ESRI et les utilisera exactement comme les pilotes :ref:`gdal.ogr.formats.pgeo` et :ref:`gdal.ogr.formats.geomedia` le font. Pour les autres bases de données MDB, toutes les tables seront présentées comme couches OGR.

Unknown interpreted text role "ref".

Unknown interpreted text role "ref".

A Comment compiler le pilote MDB (sur Linux)

Vous avez besoin d'un JDK (JRE n'est pas suffisant) pour compiler le pilote.

Sur Ubuntu 10.04 avec le packaqe *openjdk-6-jdk* installé,

```
./configure --with-java=yes --with-mdb=yes
```

Il est possible d'ajouter l'option *--with-jvm-lib-add-rpath* pour inclure le chemin dans la *libjvm.so* dans la bibliothèque GDAL.

Sur les autres version de Linux, vous pouvez devoir spécifier :

```
./configure --with-java=/path/to/jdk/root/path --with-jvm-
lib=/path/to/libjvm/directory --with-mdb=yes
```

où */path/to/libjvm/directory* est par exemple */usr/lib/jvm/java-6-openjdk/jre/lib/amd64*.

B Comment lancer le pilote MDB (sur Linux)

Vous avez besoin de JRE et 3 JAR externe pour lancer le pilote.

1. si vous n'avez pas spécifié *--with-jvm-lib-add-rpath* au moment du *configure*, définissez le chemin du répertoire qui contient *libjvm.so* dans *LD_LIBRARY_PATH* ou dans /etc/ld.so.conf.
2. Télécharger *jackcess-1.2.2.jar*, *commons-lang-2.4.jar* et *commons-logging-1.1.1.jar* (les autres versions peuvent fonctionner).
3. placer les 3 JARs soit dans le répertoire *lib/ext* du JRE (par exemple */usr/lib/jvm/java-6-openjdk/jre/lib/ext*) ou dans un autre répertoire et pointer explicitement avec la variable d'environnement *CLASSPATH*.

C Ressources

- Page principale de la bibliothèque Jackcess.
- Utilitaire qui contient les dépendances JARs nécessaires.

D Voir également

- La page du pilote :ref:`gdal.ogr.formats.pgeo`.
  Unknown interpreted text role "ref".
- La page du pilote :ref:`gdal.ogr.formats.geomedia`
  Unknown interpreted text role "ref".

Chapitre CXVI Memory

Le pilote implémente l'accès en lecture et en écriture des couches d'objet géométrique entièrement en mémoire. Ceci est d'abord utile pour augmenter les performances et est un espace de stockage grandement malléable. Toutes les options de mises à jour, types de géométries, et types de champ sont gérés.

Il n'y aucun moyen d'ouvrir un espace de stockage existante en mémoire. Il doit être crée avec *CreateDataSource()* et remplie puis utilisé en continuité. Lorsque l'espace de stockage est fermé, tout le contenu est libéré et détruit.

Le pilote n'implémente pas d'index attributaire ou spatial, les requêtes attributaires et spatiales sont encore évaluées pour toutes les géométries. Chercher les géométries par id des géométries devrait être rapide (juste un tableau à lire et les géométrie à copier).

A Problèmes de création

N'importe quel nom peut être utilisé pour une source de données crée. Il n'y a pas d'options de création de source de données ou de couche. Les noms des couches nécessitent d'être unique, mais n'a pas d'autres contraintes.

Les id des géométries envoyés à *CreateFeature()* sont préservé à moins qu'ils n'excèdent 10 000 000 auquel cas ils seront remis à zéro pour éviter un besoin d'un tableau excessivement large et des géométries clairsemées.

De nouveau champs peuvent être ajouté aux couches qui ont des déjà des géométries.

Chapitre CXVII MapInfo TAB et MIF/MID

Les jeux de données Mapinfo aux formats natif (TAB) ou interchangeable (MIF/MID) sont gérés en lecture et écriture. La mise à jour de fichiers existants n'est pas gérée pour le moment.

<note>Dans le reste de ce document, on utilisera l'expression "fichier MIF/MID" pour désigner une paire de fichiers .MIF + .MID, et "fichier .TAB" renverra au jeu de fichiers d'une table Mapinfo dans sa forme binaire (généralement, les fichiers .TAB, .DAT, .MAP, .ID et .IND).</note>

Le pilote Mapinfo considère un dossier entier comme un *jeu de données*, et un fichier unique dans ce dossier comme une *couche*. Le nom du dossier doit donc être choisi pour nommer le jeu de données. Il est néanmoins tout à fait possible d'utiliser le nom d'un seul fichier (.TAB ou .MIF) du dossier pour nommer le jeu de données ; ce dernier sera alors considéré comme ne comprenant qu'une seule couche.

Les informations de système de coordonnées de Mapinfo sont gérées en lecture et écriture.

A Problèmes de création

Le format des fichiers .TAB nécessite que les limites (l'extension géographique) d'un nouveau fichier soient indiquées avant l'écriture des premières caractéristiques. Il n'y a toutefois pour le moment pas de mécanisme propre pour paramétrer les limites par défaut d'un nouveau fichier avec l'interface *OGRDataSource*. Nous comptons corriger le pilote pour être en mesure d'indiquer des limites par défaut pour chaque projection, mais actuellement, le pilote Mapinfo règle les limites par défaut d'un nouveau fichier comme suit :

- Pour un fichier en coordonnées géographiques (Lat/Lon) : `LIMITES (-180, -90) (180, 90)`
- Pour toute autre projection : `LIMITES (-30000000, -15000000) (30000000, 15000000)`

Si aucun système de coordonnées n'est fourni à la création de la couche, c'est la *projection* qui est utilisée par défaut, pas le système *géographique*, ce qui peut entraîner une précision très grossière si les coordonnées sont vraiment géographiques. Vous pouvez ajouter *-a_srs WGS84* à la ligne de commande `ogr2ogr` pendant la conversion

pour forcer le mode géographique.

Les données attributaires de Mapinfo connaissent un certain nombre de limitations :

- Seuls les champs *Integer*, *Real* et *String* (chaîne de caractère) peuvent être créés. La liste variée et les champs de type binaire ne peuvent pas être créés.
- Pour les champs de type *Chaîne de caractères*, la taille du champ est utilisée pour établir la taille du stockage dans le fichier .DAT. Cela signifie que les chaînes plus longues que la longueur du champ seront tronquées.
- Les champs de type *Chaîne de caractères* sans longueur assignée sont considérés comme ayant 254 caractères.

B Options de création de jeux de données

- **FORMAT=MIF :** Pour créer des fichiers .MIF/.MID au lieu de fichiers .TAB (la sortie est en .TAB par défaut).

C Options de création de couche

- **SPATIAL_INDEX_MODE=QUICK :** A utiliser pour activer le "mode d'indexation spatiale rapide".

Le comportement par défaut de MITAB depuis GDAL v1.5.0 est de générer un index spatial optimisé par défaut, mais cela provoque une moins grande vitesse d'écriture que celle que donnaient GDAL 1.4.X et précédents. Les applications qui nécessitent une plus grande vitesse d'écriture et qui ne se soucient pas de la performance des requêtes spatiales sur le fichier de sortie peuvent utiliser cette option pour demander la création d'un index spatial non-optimal (qui émule en fait le type d'index spatial produit par le pilote .TAB d'OGR avant GDAL 1.5.0). Dans ce mode, l'écriture de fichiers peut être environ 5 fois plus rapide, mais les requêtes spatiales peuvent être jusqu'à 30 fois plus lentes.

1 Compatibilité

Avant la v1.8.0, le pilote utilisait d'une manière incorrect un "." comme délimiteur pour les paramètres id: et à partir de v1.8.0 le pilote utilise une virgule comme délimiteur comme le prévoie la spécification Feature Style d'OGR.

D Voir aussi

- MITAB

Chapitre CXVIII MSSQLSpatial - Microsoft SQL Server Spatial Database

Ce pilote implémente la gestion de l'accés aux tables spatiales dans Microsoft SQL Server 2008+ qui contiennent les types de données géométriques et géographiques pour représenter les colonnes géométrie.

A Se connecter à une base de données

Pour se connecter à un source de données MSSQL, utilisez une chaîne de connexion définissant le nom de la base de données avec des paramètres supplémentaires si nécessaire. La chaîne de connexion doit être préfixé par '*MSSQL:*'.

```
MSSQL:server=.\MSSQLSERVER2008;database=dbname;trusted_connection=yes
```

En plus des paramètres standards du format de la chaîne de connexion les paramètres personnalisés suivants peuvent également être utilisés avec la syntaxe suivante :

- **Tables=schema1.table1(geometry column1),schema2.table2(geometry column2) :**

  en utilisant ce paramètre vous pouvez définir le sous jeu de données des couches à utiliser par le pilote. Si ce paramètre n'est pas définie, les couches sont récupérées à partir de la table de métadonnées *geometry_column*. vous pouvez omettre de spécifier la partie sur le schéma et la colonne géométrique de la syntaxe.
- **GeometryFormat=native|wkb|wkt :**

  Le format désiré dans lequel les géométries doivent être récupérées à partir du serveur. La valeur par défaut est '*native*' dans ce cas le format de sérialisation natif SqlGeometry et SqlGeography est utilisé. Lors de l'utilisation de '*wkb*' ou '*wkt*' la représentation de la géométrie est convertie en '*Well Known Binary*' et '*Well Known Text*' dans le serveur. Cette conversion nécessite une surcharge significative côté serveur et rend l'accès aux features plus lente qu'en utilisant le format natif.

Les noms des paramètres ne sont pas sensible à la casse dans la chaîne de connexion. La définition du paramètre **Database** est nécessaire pour le pilote afin de sélectionner la base de données correctement.

B Requêtes SQL

Le pilote MS SQL Spatial envoie les requêtes SQL directement à MS SQL par défaut, plutôt que de les évaluer en interne en utilisant l'appel à *ExecuteSQL()* du *OGRDataSource*, ou l'option de la commande *-sql* d'`ogr2ogr`. Les expressions des requêtes attributaires sont aussi envoyé directement à MSSQL. Il est aussi possible de demander au pilote MSSQL d'OGR de prendre en charge les commandes SQL avec le moteur :ref:`gdal.ogr.sql`, en passant la chaîne **"OGRSQL"** à la méthode *ExecuteSQL()*, comme le nom du dialect SQL.

Unknown interpreted text role "ref".

Le pilote MSSQL dans OGR gère les appels *OGRLayer::StartTrasaction()*, *OGRLayer::CommitTransaction()* et *OGRLayer::RollbackTransaction()* dans le sens normal de SQL.

C Problèmes lors de la création

Ce pilote ne gère pas la création d'une nouvelle base de données, vous devrez utiliser l'*Outils client du Serveur Microsoft SQL* pour cela, mais il permet la création de nouvelles couches dans une base existante.

1 Options de création de couche

- **GEOM_TYPE**: l'option de création *GEOM_TYPE* peut être définie parmi "geometry" ou "geography". Si cette option n'est pas définie la valeur par défaut est "geometry". Pour créer la colonne géométrie de type "geography", ce paramètre doit être définie en "geography". Dans ce cas, la couche doit avoir une référence spatiale valide de l'un des systèmes de coordonnées géographiques définies dans la table de métadonnées du serveur SQL **sys.spatial_reference_systems**. Les systèmes de coordonnées projetées ne sont pas pris en charge dans ce cas.
- **OVERWRITE**: doit être à "YES" pour forcer une couche existante au nom définie à être détruite avant la création de la couche demandée.
- **LAUNDER**: doit être à "YES" pour forcer le nettoyage des nouveaux champs créés dans cette couche dans une forme compatible avec MSSQL. Cela les convertie en minuscule et certains caractères spéciaux comme "-" et "#" en "_". Si "NO" les noms exactes sont préservés. La valeur par défaut est "YES". Si activé le nom de la table (couche) sera aussi nettoyer.
- **PRECISION**: doit être à "YES" pour forcer les nouveaux champs créés sur cette couche de tenter et de représenter les informations sur la largeur et la précision, si disponible en utilisant les types *numeric(width,precision)* ou *char(width)*. Si "NO" alors les types *float*, *int* et *varchar* seront utilisé. La valeur par défaut est "YES".
- **DIM={2,3}**: contrôle la dimension de la couche. 3 par défaut.
- **GEOM_NAME**: définie le nom de la colonne géométrique dans la nouvelle table. si omis par défaut *ogr_geometry*.
- **SCHEMA**: définie le nom du schéma pour la nouvelle table. Si ce paramètre n'est pas géré le schéma par défaut est "*dbo*".

- **SRID**: définie l'id de la référence spatiale de la nouvelle table explicitement. L'entrée correspondante doit déjà existé dans la table de métadonnées *spatial_ref_sys*. Si ce paramètre n'est pas définie le SRID est dérivé à partir du code d'autorité du SRS de la couche source.

2 Création d'index spatial

Par défaut le pilote MS SQL Spatial ne créé par d'index spatiaux pour la table pendant la création de la couche. Cependant vous devez créer un index spatial en utilisant l'option sql suivante :

```
create spatial index on schema.table
```

L'index spatial peut aussi être supprimé en utilisant la syntaxe suivante :

```
drop spatial index on schema.table
```

D Exemples

Créer une couche à partir d'un source de données OGR :

```
ogr2ogr -overwrite -f MSSQLSpatial
"MSSQL:server=.\MSSQLSERVER2008;database=geodb;trusted_connection=yes"
"rivers.tab"
```

Se connecter à une couche et dumper son contenu :

```
ogrinfo -al
"MSSQL:server=.\MSSQLSERVER2008;database=geodb;tables=rivers;trusted_c
onnection=yes"
```

Créer un index spatial :

```
ogrinfo -sql "create spatial index on rivers"
"MSSQL:server=.\MSSQLSERVER2008;database=geodb;trusted_connection=yes"
```

Chapitre CXIX MySQL

Le pilote implémente l'accès en lecture et écriture pour les données spatiale dans les tables MySQL. Cette fonctionnalité a été introduite dans GDAL/OGR 1.3.2.

Lors de l'ouverture d'une base de données, son nom doit être définie sous la forme "MYSQL:dbname[,options]" où *options* peut inclure des informations séparées par des virgules comme "user=*userid*", "password=*password*", "host=*host*" et "port=*port*".

De même, une option "tables=*table*;*table*..." peut être ajouté pour restreindre l'accès à une liste spécifique de tables dans la base de données. Cette option est d'abord utile lorsqu'une base de données contient beaucoup de tables, et scanner tout le schéma prendrait beaucoup de temps.

Pour l'instant toutes les tables habituelles sont supposées être des couches d'un point de vue d'OGR., avec les noms des tables comme nom de couche. Les vues nommées ne sont pas gérées pour l'instant.

Si un champ d'entier simple est une clé primaire, il sera utilisé comme FID autrement le FID sera assigné séquentiellement et la recherche par FID sera extrêmement lente.

Par défaut, les requêtes SQL sont passé directement au moteur de base de données MySQL. Il est également possible de questionner le pilote pour prendre en charge les commandes SQL avec le [[ogr_sql|moteur SQL d'OGR]], en passant la chaine "OGRSQL" à la méthode *ExecuteSQL()*, en tant que nom du dialecte SQL.

A Avertissements

- Dans le cas d'une couche définie par une commande SQL, les champs soit qui sont nommés "OGC_FID" soit ceux qui sont définie NOT NULL, ceux qui sont une clé primaire (*PRIMARY KEY*), et ceux qui sont un champ de type entier seront supposé être un FID.
- Les champs géométriques sont lu à partir de MySQL en utilisant le format WKB. Les version plus ancienne que 5.0.16 de MySQL sont connus pour avoir un problème avec certaine génération de WKB et peuvent ne pas fonctionner proprement.
- La colonne OGR_FID, qui peut être écrasée avec l'option de création de couche MYSQL_FI, est implémenté comme un champ *INT UNIQUE NOT NULL*

AUTO_INCREMENT. Cela semble créer implicitement un index sur le champ.

- la colonne géométrique, par défaut à *SHAPE* et qui peut être écrasée avec l'option de création de couche *GEOMETRY_NAME*, est crée comme une colonne NOT NULL à moins que SPATIAL_INDEX soit désactivé. Par défaut, un index spatial est crée au moment de la création de la table.
- L'information SRS est stocké en utilisant *Simple Features* de l'OGC, avec la création des tables de méta-données *geometry_columns* et *spatial_ref_sys* dans la base de données définies si elles ne le sont pas déjà. La table *spatial_ref_sys* n'est pas pré-rempli avec les valeurs EPSG et SS comme PostGIS. Si aucun code EPSG n'ait présent pour une table donnée, la valeur MAX(SRID) sera utilisée.
- Le timeouts de la connection au serveur peut être définie par la variable d'environnement*MYSQL_TIMEOUT*. Par exemple, *SET MYSQL_TIMEOUT=3600*. Il est possible que cette variable ait un impact seulement quand le système du serveur MySQL est Windowss.
- Le pilote MySQL ouvre une connection à la base de données en utilisant le mode *CLIENT_INTERACTIVE*. Vous pouvezajuster ce paramètre (*interactive_timeout*) dans le fichier *mysql.ini* ou *mysql.cnf* de votre serveur.
- Nous utilisons le WKT pour insérer les géométries dans la base de données. Si vous insérez de grosse géométrie, vous devez faire attention du paramètre *max_allowed_packet* dans la configuration MySQL. Par défaut, il est définie à 1 M, mais cela peut ne pas être asse important pour les très grosses géométries. Si vous obtenez un message d'erreur du type : *Got a packet bigger than 'max_allowed_packet' bytes*, vous devez augmenter ce paramètre.

B Problèmes de création

Le pilote MySQL ne gère pas la création de nouveaux jeu de données (une base de données dans MySQL), mais il permet la création de nouvelles couches dans une base existante.

Par défaut, le pilote MySQL tentera de préserver la précision des géométries OGR lors de la création et la lecture des couches MySQL. Pour les champs d'entier avec une longueur définie, il utilisera DECIMAL comme type de champ MySQL avec une précision définie de 0. Pour les champs de type réel, il utilisera DOUBLE de largeur et précision définie. Pour les champs caractères avec une largeur définie, VARCHAR sera utilisé.

Le pilote MySQL n'essaye pas de comprendre l'encodage des caractères pour le moment.

Le pilote MySQL n'est pas transactionnel pour le moment.

C Options de création de couche

- **OVERWRITE :** elle peut être définie à "YES" pour forcer les couches existantes de même nom à être détruite avant la création de la couche demandée.
- **LAUNDER :** elle peut être définie à "YES" pour forcer les champs crées à avoir leur nom traduit sous une forme plus compatible avec MySQL. Cela les convertit en minuscule et certains caractères spéciaux comme "-" et "#" en "_". Si "NO" les noms exacts sont préservés. La valeur par défaut est "YES".
- **PRECISION :** elle peut être à "TRUE" pour tenter de préserver la largeur et la précision des champs lors de la création et la lecture des couches MySQL. La valeur par défaut est "TRUE".
- **MYSQL_GEOM_COLUMN :** cette option définie le nom de la colonne géométrique. La valeur par défaut est "SHAPE".
- **MYSQL_FID :** cette option définie le nom de la colonne FID. La valeur par défaut

est "OGR_FID"

- **SPATIAL_INDEX :** elle peut être définie "NO" pour arrêter la création automatique d'un index spatial sur la colonne géométrique, permettant des géométries NULL et probablement un chargement plus rapide.
- **ENGINE :** définie optionnellement le moteur de la base de données à utiliser. Pour MySQL 4.x cela doit être définie à MyISAM pour les tables spatiales.

L'exemple suivant du nom de la source de données ouvre le schéma *westholland* de la base de données dont le mot de passe est *psv9570* pour l'utilisateur *root* sur le port 3306. Aucune nom d'hôte n'est fournit, donc localhost est utilisé. La directive *table=* est scanné et présenté comme la couche à utiliser.

```
MYSQL:westholland,user=root,password=psv9570,port=3306,tables=bedrijve
n
```

L'exemple suivant utilise ogr2ogr pour créer une copie de la couche des frontières mondiales à partir d'une shapefile dans une table MySQL. Il écrase la table *borders2* existante, définie une option lors de la création pour définir le nom de la colonne géométrique *SHAPE2*.

```
ogr2ogr -f MySQL MySQL:test,user=root world_borders.shp -nln borders2
-update -overwrite -lco GEOMETRY_NAME=SHAPE2
```

L'exemple suivant utilise ogrinfo pour renvoyer des informations synthétique sur la couche *borders2* dans la base de données *test* :

```
ogrinfo MySQL:test,user=root borders2 -so

    Layer name: borders2
    Geometry: Polygon
    Feature Count: 3784
    Extent: (-180.000000, -90.000000) - (180.000000, 83.623596)
    Layer SRS WKT:
    GEOGCS["GCS_WGS_1984",
        DATUM["WGS_1984",
            SPHEROID["WGS_84",6378137,298.257223563]],
        PRIMEM["Greenwich",0],
        UNIT["Degree",0.01745329251994295]]
    FID Column = OGR_FID
    Geometry Column = SHAPE2
    cat: Real (0.0)
    fips_cntry: String (80.0)
    cntry_name: String (80.0)
    area: Real (15.2)
    pop_cntry: Real (15.2)
```

Chapitre CXX NAS - ALKIS

Le lecteur NAS lit le format NAS/ALKIS utilisé pour les données cadastrales en Allemagne. Le format est un profile GML avec des objets GML3 complexe qui ne peuvent pas être lu facilement avec le pilote GML générique d'OGR.

Le pilote dépend de la compilation de GDAL/OGR avec la bibliothèque XML Xerces.

Ce pilote a été implémenté dans le contexte du projet PostNAS qui a plus d'information sur son utilisation.

A Voir également

- PostNAS

Chapitre CXXI UK .NTF

Le Format de Transfert National (*National Transfer Format*)), principalement utilisé par le *UK Ordnance Survey*, est géré en lecture seule.

Ce pilote traite un répertoire comme un jeu de données et tente de fusionner tous les fichiers .NTF du répertoire, produisant une couche pour chaque type de géométrie (mais généralement pas pour chaque fichier source). Un répertoire contenant plusieurs fichiers *landlines* aura donc 3 couches (*LANDLINE_POINT*, *LANDLINE_LINE* et *LANDLINE_NAME*) sans regard du nombre de fichier *landline*.

Les géométries NTF sont toujours envoyées avec le système de coordonnées *British National Grid*. Cela peut être inapproprié pour les fichiers NTF écrit par d'autre organisation que le *UK Ordnance Survey*.

Voir également :

- Information Générale sur le NTF UK

A Implementation Notes

1 Produits (et couches) gérés

- **Landline (et Landline Plus) :**
  - *LANDLINE_POINT*
  - *LANDLINE_LINE*
  - *LANDLINE_NAME*
- **Panorama Contours :**
  - *PANORAMA_POINT*
  - *PANORAMA_CONTOUR*
  - attribut *HEIGHT* gère l'élévation.
- **Strategi :**
  - *STRATEGI_POINT*
  - *STRATEGI_LINE*
  - *STRATEGI_TEXT*
  - *STRATEGI_NODE*
- **Meridian :** * *MERIDIAN_POINT* * *MERIDIAN_LINE* * *MERIDIAN_TEXT* * *MERIDIAN_NODE*

- **Boundaryline :**
  - *BOUNDARYLINE_LINK*
  - *BOUNDARYLINE_POLY*
  - *BOUNDARYLINE_COLLECTIONS*
  - La couche *_POLY* a des liens vers des liens permettant les polygones réels d'être formés (autrement les couches *_POLY* seulement on un point nécessaire pour la géométrie). Les collections sont des collections de polygones (également sans géométrie comme lu). C'est le seul produit à partir duquel les polygones peuvent être construit.
- **BaseData.GB :**
  - *BASEDATA_POINT*
  - *BASEDATA_LINE*
  - *BASEDATA_TEXT*
  - *BASEDATA_NODE*
- **OSCAR Asset/Traffic :**
  - *OSCAR_POINT*
  - *OSCAR_LINE*
  - *OSCAR_NODE*
- **OSCAR Network :**
  - *OSCAR_NETWORK_POINT*
  - *OSCAR_NETWORK_LINE*
  - *OSCAR_NETWORK_NODE*
- **Address Point :**
  - *ADDRESS_POINT*
- **Code Point :**
  - *CODE_POINT*
- **Code Point Plus :**
  - *CODE_POINT_PLUS*

Le jeu de données en entier possèdera également une couche *FEATURE_CLASSES* contenant une table vierge relatant les nombres *FEAT_CODE* avec les noms des classes d'objet (*FC_NAME*). Ceci s'applique à tous les produits dans le jeu de données. Quelques types de couche (tel que les produits *Code point* et *Address Point*) n'incluent pas de classes d'objet. Certain produits utilisent des classes d'objets qui ne sont pas définie dans le fichier, et ils n'apparaitront donc pas dans la couche *FEATURE_CLASSES*.

2 Schémas des produits

L'approche entreprise pour ce lecteur est de traiter un fichier, ou un répertoire de fichier comme un simple jeu de données. Tous les fichiers dans le jeu de données sont scannés à l'ouverture. Pour chaque produit particulier (listé au dessus) un ensemble de couches est créé ; cependant ces couches peuvent être extraites de plusieurs fichiers du même produit.

Les couches sont basées sur un type d'objet de faible niveau dans le fichier NTF, mais contiendront généralement des objets de plusieurs codes d'objet (attribut *FEAT_CODE*). Différents objets dans une couche données peuvent avoir une variété d'attribut dans le fichier ; cependant le schéma est établit en fonction de l'union de tous les attributs possible des objets d'un type particulier (par exemple les points) de cette famille de produit (par exemple le réseau OSCAR).

Si un produit NTF lu ne correspondant pas à un des schéma connu il sera géré par un parseur générique différent qui gère seulement des couches de type *GENERIC_POINT* et *GENERIC_LINE*. Seul l'objet aura un attribut *FEAT_CODE*.

Les détails sur quelles couches de quels produits ont quels attributs peuvent être trouvés dans la méthode *NTFFileReader::EstablishLayers()* à la fin du fichier *ntf_estlayers.cpp*. Ce fichier contient également tous les codes de traduction spécifique au produit.

3 Attributs spéciaux

- **FEAT_CODE :** code générale entier de l'objet, peut être utilisé pour rechercher un nom dans la table/couche *FEATURE_CLASSES*.
- **TEXT_ID/POINT_ID/LINE_ID/NAME_ID/COLL_ID/POLY_ID/GEOM_ID :** identifiant unique pour un objet de type approprié.
- **TILE_REF :** Toutes ces couches (sauf *FEATURE_CLASSES*) contiennent un attribut *TILE_REF* qui indique de quelle tuile (fichier) provient l'objet. De manière générale les numéros d'id sont seulement unique dans la tuile et dont *TILE_REF* peut être utilisé pour restreindre les liens des id dans les objets du même fichier.
- **FONT/TEXT_HT/DIG_POSTN/ORIENT :** Des informations détaillées sur la police, la hauteur du texte, la position de digitalisation, et l'orientation du texte ou les noms des objets. Lisez le manuel du produit OS pour comprendre les unités, la signification de ces codes.
- **GEOM_ID_OF_POINT :** Pour les objets *_NODE* cela définie le *POINT_ID* de l'objet de la couche ponctuelle auquel ce nœud correspond. De manière générale les nœuds ne portent pas une géométrie eux-même. Le nœud doit être lié à un point pour établir sa position.
- **GEOM_ID_OF_LINK :** une liste d'objet *_LINK* ou *_LINE* pour démarrer ou finir à un nœud. Les nœuds et ce champ ont généralement seulement une valeur lors de l'établissement de la connection des objets lignes pour l'analyse de réseau. Notez que cela doit être lié à l'objet cible *GEOM_ID*, et pas à son *LINE_ID*. Sur la couche *BOUNDARYLINE_POLY* cet attribut contient le *GEOM_ID* des lignes qui forme un contour de polygone.
- **POLY_ID :** un liste de *POLY_ID* de la couche *BOUNDARYLINE_POLY* associée avec une collection donnée dans la couche *BOUNDARYLINE_COLLECTIONS*.

4 Produits génériques

Dans le cas où un fichier n'est pas identifier comme faisant partie d'un produit connus existant il sera traité d'une manière générique. Dans ce cas le jeu de donnés complet est scanné pour établir quels objets ont quels attributs. À cause de cela, ouvrir un jeu de données générique peut être beaucoup plus lent qu'ouvrir un jeu de données reconnus. En se basant sur ce scan une liste d'objet générique (couches) est définie à partir de l'ensemble suivant :

```
GENERIC_POINT
GENERIC_LINE
GENERIC_NAME
GENERIC_TEXT
GENERIC_POLY
GENERIC_NODE
GENERIC_COLLECTION
```

Les produits génériques sont d'abord pris en charge par le module *ntf_generic.cpp* tandis que les produits spécifiques dans *ntf_estlayers.cpp*.

Parce qu'on a trouvé des produits de données (des jeux de données OSNI) ne provenant pas de l'*Ordnance Survey* ayant des groupes d'enregistrement dans un ordre inhabituel

comparé à ce que fait l'*Ordnance Survey* anglais, il a été nécessaire de mettre en cache tous les enregistrements des produits génériques de niveau 3 et au dessus, et de construire des groupes d'enregistrement par référence d'identifiant à partir de ce cache plutôt que de dépendre d'un ordre d'enregistrement pratique. Cela est accomplit par la capacité d'indexage de *NTFFileReader* quasiment à la fin du fichier *ntffilereader.cpp*. À cause de ce cache en mémoire Les jeux de données génériques qui accèdent à l'indexage peuvent demander plus de mémoire qu'accéder à des produits de données connus, bien qu'il ne soit pas nécessaire pour les produits générique de niveau 1 et 2.

Il est possible de forcer un produit connus pour qu'il soit traité comme générique en définissant l'option *FORCE_GENERIC* à *ON* en utilisant *OGRNTFDataSource::SetOptionsList()* comme il est indiqué dans le fichier *ntfdump.cpp*. Cela peut également être accomplit en dehors des applications *OGR* en définissant la variable d'environnement *OGR_NTF_OPTIONS* à "*FORCE_GENERIC=ON*".

Chapitre CXXII Oracle Spatial

Ce pilote gère la lecture et l'écriture de données dans le format objet-relationnel d'Oracle Spatial (8.1.7 et plus récent). Le pilote d'Oracle Spatial n'est pas compilé par défaut dans OGR, mais peut l'être sur les plateformes où les bibliothèques cliente d'Oracle sont disponible.

Lors de l'ouverture d'une base de données, son nom doit être définie sous la forme *<<OCI:userid/password@database_instance:table,table>>*. La liste des tables est optionnelle. La partie *database_instance* peut être omise lors de l'accès à l'instance de base de données local par défaut.

Si la liste des tables n'est pas fournie, alors toutes les tables apparaissant dans la table *ALL_SDO_GEOM_METADATA* seront traité par OGR comme des couches avec les noms de table comme nom de couche. Les tables non-spatiales ou les tables spatiales non listé dans la table *ALL_SDO_GEOM_METADATA* ne sont pas accessible à moins de les lister dans le nom de la source de données. Même dans des bases de donnés où toutes les couches désirées sont dans la table *ALL_SDO_GEOM_METADATA*, il peut être préférable de lister seulement les tables à utiliser puisque cela peut réduire substantiellement le temps d'initialisation dans les bases de données avec beaucoup de tables.

Si la table à une colonne de type entier appelée *OGR_FID* il sera utilisé comme identifiant d'objet par OGR (et il n'apparait pas comme un attribut normal). Lors du chargement des données dans Oracle Spatial, OGR créera toujours le champ *OGR_FID*.

A Problèmes avec SQL

Par défaut le pilote d'Oracle envoie une requête SQL directement au moteur Oracle plutôt que de l'évaluer en interne lors de l'utilisation de l'appel *ExecuteSQL()* sur *OGRDataSource*, ou l'option de commande *-sql* à `ogr2ogr`. Les expressions de requête d'attributs sont également envoyé à Oracle.

De même deux commandes spéciales sont gérées via l'interface *ExecuteSQL()*. Ce sont "*DELLAYER:<table_name>*" pour supprimer une couche et "*VALLAYER:<table_name>*" pour appliquer une vérification *SDO_GEOM.VALIDATE_GEOMETRY()* sur la couche. En interne, ces pseudo-commandes sont traduit en commandes plus complexe pour Oracle.

Il est également possible de demander au pilote de prendre en charge les commandes SQL avec le :ref:`gdal.ogr.sql`, en envoyant la chaine "*OGRSQL*" à la méthode

ExecuteSQL() comme nom du dialecte SQL.

Unknown interpreted text role "ref".

B Avertissements

- La logique de reconnaissance de type est actuellement assez pauvre. Aucun effort n'est fait pour préserver la longueur réelles pour les champs de type entier et réel.
- Différents types tels que les objets et les BLOBs dans Oracle seront complètement ignorés par OGR.
- Actuellement les sémantiques de transaction d'OGR ne sont pas proprement liées aux sémantiques de transaction dans Oracle.
- Si un attribut appelée *OGR_FID* existe dans le schéma pour les tables lues, il sera utilisé comme FID. Les lectures aléatoires (basé sur le FID) sur des tables sans champ FID identifié (et indexé) peut être très lente. Pour forcer l'utilisation d'un nom de champ particulier la variable de configuration *OCI_FID* (c'est à dire une variable d'environnement) peut être définie par le nom du champ cible.
- Les types de géométries courbes sont convertie en *linestrings* ou en *anneau linéaire* en segments de 6 degrés lors de la lecture. Le pilote ne gère pas l'écriture de géométries courbes.
- Il n'y a pas de gestion pour les nuages de point(SDO_PC), TIN (SDO_TIN) et les types de données de texte d'annotation dans Oracle Spatial.

C Problèmes de création

Le pilote Oracle Spatial ne gère pas la création de nouveaux jeux de données (instances de bases de données), mais il doit permettre la création de nouvelles couches dans une base de données existantes.

Dès la fermeture de *OGRDataSource* les nouvelles couches créées auront un index spatial automatiquement créé. À ce moment la table *USER_SDO_GEOM_METADATA* sera également mise à jour avec les limites pour la table basé sur les objets qui ont été réellement écris. Une conséquence de cela est qu'une fois que une couche a été chargée il n'est généralement pas possible de charger des objets additionnel en dehors de l'étendue originel sans modifier manuellement l'information *DIMINFO* dans *USER_SDO_GEOM_METADATA* et reconstruire l'index spatial.

1 Options de création de couche

- **OVERWRITE :** peut être *YES* pour forcer une couche existante au nom désiré d'être détruite avant la création de la couche demandée.
- **TRUNCATE :** peut être à "YES" pour forcer la table existante pour être réutilisée, mais en vidant la table, préservant les indexes ou les dépendances.
- **LAUNDER :** peut être *YES* pour forcer les nouveaux champs créés sur cette couche à avoir leurs noms de champ "nettoyer" dans une forme plus compatible avec Oracle. Cela convertit les lettres en minuscule et certains caractères spéciaux comme - et # en _. La valeur par défaut est *NO*.
- **PRECISION :** peut être *YES* pour forcer de nouveaux champs à créer sur cette couche pour essayer et représenter l'information de la largeur et de la précision, en utilisant les types *NUMBER(largeur,précision)* ou *VARCHAR2(largeur)* si disponible. Si *NO* alors les les types *NUMBER*, *INTEGER* et *VARCHAR2* seront utilisés à la place. La valeur par défaut est *YES*.

- **DIM :** peut être définie à 2 ou 3 pour forcer la dimension de la couche créée. S'il n'est pas définie, 3 est utilisé par défaut.
- **INDEX :** peut être définie à *OFF* pour désactiver la création d'un index spatial quand une couche est chargé complètement. Par défaut un index est créé si n'importe quel objet d'une couche possède des géométries valides.
- **INDEX_PARAMETERS :** peut être utilisé pour passer des paramètres de création lorsque l'index spatial est créé. Par exemple définir *INDEX_PARAMETERS* à *SDO_LEVEL=5* entrainera l'utilisation d'un index à tuile de 5 niveaux. Par défaut aucun paramètre n'est passé entrainant la création d'un index spatial R-Tree.
- **DIMINFO_X :** peut être définie aux valeurs xmin,xmax,xres pour contrôler l'information de la dimension X écrite dans la table *USER_SDO_GEOM_METADATA*. Par défaut les étendues sont collectées à partir des données écrites réelles.
- **DIMINFO_Y :** peut être définie aux valeurs ymin,ymax,yres pour contrôler l'information de la dimension Y écrite dans la table *USER_SDO_GEOM_METADATA*. Par défaut les étendues sont collectées à partir des données écrites réelles.
- **DIMINFO_Z :** peut être définie aux valeurs zmin,zmax,zres pour contrôler l'information de la dimension Z écrite dans la table *USER_SDO_GEOM_METADATA*. Par défaut les valeurs fixées de -100000,100000,0.002 sont utilisées pour les couches avec une 3e dimension.
- **SRID :** Par défaut, ce pilote tentera de trouver une ligne existante dans la table *MDSYS.CS_SRS* avec un système de coordonnées au format Well known Text correspondant exactement à celui du jeu de données. Si aucun n'est trouvé, une nouvelle ligne sera ajoutée à cette table. L'option de création de SRID permet aux utilisateurs de forcer l'utilisation d'un SRID Oracle existant même s'il ne correspond pas exactement au WKT auquel le pilote s'attend.
- **MULTI_LOAD :** Si activé les nouveaux objets seront créés en groupe de 100 par commande INSERT SQL, au lieu que chaque objet soit une commande INSERT séparée. En ayant cela activé est le moyen le plus rapide de charger des données rapidement. Le mode multi-load est activé par défaut, et peut être forcé pour ne pas l'être pour les couches existantes ou pour les nouvelles couches en la définissant à *NO*.
- **LOADER_FILE :** Si cette option est définie, toutes les informations des objets seront écrites dans un fichier utilisable avec le chargeur SQL au lieu d'être insérées directement dans la base de données. La couche en elle-même est toujours créée dans la base de données immédiatement. La gestion du chargeur SQL est encore expérimentale, et généralement le mode *MULTI_LOAd* activé doit être utilisé à la place lors des essaies pour des performances optimales des chargements.
- **GEOMETRY_NAME :** Par défaut OGR créé de nouvelles tables avec une colonne géométrique nommé *ORA_GEOMETRY*. Si vous préférez utiliser un nom différent, il peut être fournit avec l'option de création de couche *GEOMETRY_NAME*.

2 Exemple

Simple traduction d'un shapefile vers Oracle. La table 'ABC' sera créée avec les objets provenant du fichier *abc.shp* et les attributs du fichier *abc.dbf*.

```
% ogr2ogr -f OCI OCI:warmerda/password@gdal800.dreadfest.com abc.shp
```

Ce second exemple charge une couche des frontières politique à partir d'un VPF (avec le [[ogr_ogdi|pilote OGDI]]), et renomme la couche à partir de la couche mystérieuse d'OGDI en quelque chose de plus compréhensible. Si une table existante au nom désiré existe elle sera écrasée.

```
% ogr2ogr  -f OCI OCI:warmerda/password \
    gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia \
    -lco OVERWRITE=yes -nln polbndl_bnd 'polbndl@bnd(*)_line'
```

Cet exemple montre l'utilisation d'`ogrinfo` pour évaluer une commande de requête SQL dans Oracle. Des requêtes spécifique sophistiquées d'Oracle Spatial peuvent également être utilisé via l'option en ligne de commande *-sql* d'`ogrinfo`.

```
ogrinfo -ro OCI:warmerda/password -sql "SELECT pop_1994 from canada
where province_name = 'Alberta'"
```

D Crédits

Le développeur voudrait remercier la société SRC, LLC pour son apport financier au développement de pilote.

Chapitre CXXIII ODBC RDBMS

OGR gère en option les tables spatiales et non spatiales accédé via ODBC. ODBC est une couche d'accès générique pour l'accès à plusieurs systèmes de bases de données, et les données qui peuvent être représentée sous forme de base de données (collection de tables). La gestion d'OBDC est potentiellement disponible sous les plateformes Unix et Windows, mais est seulement inclus dans les compilations Unix par des options spéciales de configuration.

Les sources de données sont accédé en utilisant le nom de la source de données sous la forme *ODBC:userid/password@dsn,schema.tablename(geometrycolname),...:srs_tablename(srid column,srtextcolumn)*. Avec des paramètres optionnels comme ceux qui suivent sont aussi disponible :

- *ODBC:userid/password@dsn*
- *ODBC:userid@dsn,table_list*
- *ODBC:dsn,table_list*
- *ODBC:dsn*
- *ODBC:dsn,table_list:srs_tablename*

Le dsn est le nom de la source de données d'ODBC. Normalement les sources de données ODBC sont configurées en utilisant un outil d'Administration d'ODBC et assigne un DSN. Ce DSN est ce qui est utilisé pour accéder à la source de données.

Par défaut ODBC recherche la table *GEOMETRY_COLUMNS*. Si elle est trouvée elle est utilisée pour identifier l'ensemble des tables spatiales qui doivent être traitées comme couches par OGR. Si elle n'est pas trouvée, alors toutes les tables dans la source de données sont renvoyé comme des couches non spatiales. Cependant, si une liste de table (une liste séparé par des virgules de noms de table) est fournie, alors seule ces tables seront représentées comme couches (non-spatial). Cherchez l'ensemble des définitions de toutes les tables dans une base de données complexe peut être consommatrice de temps, la possibilité de restreindre l'ensemble des tables à accéder est donc d'abord un problème de performance.

Si la table *GEOMETRY_COLUMNS* est trouvée, elle est utilisée pour sélectionner la colonne pour la source de la géométrie. Si les tables sont passées dans le nom de la source de données, alors la colonne géométrique associée avec une table peut être inclus entre parenthèses après le nom de la table. Il y a pour l'instant l'hypothèse codée en dur que la géométrie est au format Well Known Binary (WKB) si le champ est binaire, ou Well

Known Text (WKT) sinon. La table *GEOMETRY_COLUMNS* doit avoir au moins les colonnes *F_TABLE_NAME*, *F_GEOMETRY_COLUMN* et *GEOMETRY_TYPE*.

Si la table a une colonne géométrique, et possède des champs appelés *XMIN*, *YMIN*, *XMAX* et *YMAX* alors les requêtes directes sur la table avec un filtre spécial accélère la requête spatiale. Les champs *XMIN*, *YMIN*, *XMAX* et *YMAX* doivent représenter l'étendue de la géométrie de la ligne dans le système de coordonnées de la table.

Par défaut, les commandes SQL sont envoyées directement au moteur de la base de données sous-jacent. Il est également possible de lancer une requête au pilote pour prendre en charge les commandes SQL avec le moteur :ref:`gdal.ogr.sql`, en passant la chaine "OGRSQL" à la méthode *ExecuteSQL()*, comme nom du dialecte SQL.

Unknown interpreted text role "ref".

A Problèmes de création

Pour l'instant le pilote ODBC d'OGR est ne lecture seule, les applications utilisant OGR ne peuvent donc pas créer de nouvelles géométries, tables et sources de données. Cette limitation sera supprimé dans le futur.

B Lisez également

- Référence de l'API ODBC MSDN <http://msdn.microsoft.com/en-us/library/ms714562(VS.85).aspx">MSDN ODBC API Reference>

Chapitre CXXIV OGDI Vectors

La gestion des vecteurs OGDI est optionnel dans OGR, et est normalement seulement configuré si OGDI est installé sur le système de compilation. S'il est disponible les vecteurs OGDI sont gérés pour l'accès en lecture pour les types de familles suivantes :

- *Point*
- *Line*
- *Area*
- *Text* (pour l'instant renvoie des points avec le texte dans l'attribut "texte")

OGDI peut (en autre formats) lire les produits VPF, tels que DCW et VMAP.

Si une URL GLTP d'OGDI est ouverte directement les possibilités OGDI 3.1 pour le pilote/serveur sont nécessaire pour obtenir une liste des couches. Une couche OGR est crée pour chaque famille OGDI disponible pour chaque couche dans le stockage de données. Pour les pilotes tel que VRF cela peut entrainer de nombreuses couches. Chacune des couches possède un nom sous OGR basé sur le nom sous OGDI plus un underscore et le nom de la famille. Par exemple une couche peut être appelée **watrcrsl@hydro(*)_line** si elle provient du pilote VRF.

À partir de GDAL/OGR 1.8.0, définir l'option de configuration *OGR_OGDI_LAUNDER_LAYER_NAMES* (ou la variable d'environnement) à YES entraînera la simplification des noms de couche. Par exemple : *watrcrsl_hydro* au lieu de 'watrcrsl@hydro(*)_line'.

Autrement pour accéder à toutes les couches du stockage de données, il est possible d'ouvrir une couche particulière en utilisant un nom de fichier personnalisé consistant à une URL GLTP correcte à laquelle vous ajoutez le nom de la couche suivit du type de la famille (séparé par une virgule). Ce mécanisme doit être utilisé pour accéder aux couches des pilotes Pre-OGDI 3.1 puisque avant OGDI 3.1 il n'y avait pas de manière correcte de découvrir les couches disponibles dans OGDI.

```
gltp:
[//<hostname>]/<driver_name>/<dataset_name>:<layer_name>:<fam
ily>
```

où <layer_name> est le nom de la couche OGDI, et <family> est choisit parmi : "line", "area", "point", ou "text".

Les informations du système de coordonnées sont gérées pour la plupart des systèmes de coordonnées. Un message sera produit quand une couche est ouverte et si le système de coordonnées ne peut être traduite.

Il n'y a pas de gestion de création ou de mise à jour dans le pilote OGDI.

Les couches Raster ne peuvent pas être lu avec ce pilote mais peut être récupéré avec le pilote Raster OGDI de GDAL.

A Exemples

Exemple d'usage d'`ogrinfo` :

```
ogrinfo gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia
'watrcrsl@hydro(*)_line'
```

Dans le nom du jeu de données *gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia* la partie *gltp:/vrf* n'est pas réellement dans le système de fichier, mais doit être ajouté. Les données VPF était à */usr4/mpp1/v0eur/*. Le répertoire *eurnasia* doit être au même niveau que les fichiers dht. et lat.. La référence *hydro* est un sous-répertoire de *eurnasia/* où *watrcrsl*.* est trouvé.

Exemple d'utilisation de conversion VMAP0 en SHAPE avec ''ogr2ogr'' :

```
ogr2ogr watrcrsl.shp gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia
'watrcrsl@hydro(*)_line'
ogr2ogr polbnda.shp  gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia
'polbnda@bnd(*)_area'
```

Une requête SQL d'OGR sur un jeu de données VMAP. Là encore, notez soigneusement les guillemets du nom de la couche :

```
ogrinfo -ro gltp:/vrf/usr4/mpp1/v0noa/vmaplv0/noamer \
       -sql 'select * from "polbndl@bnd(*)_line" where use=26'
```

B Voir également

- OGDI.SourceForge.Net
- Couvertures VMap0

Chapitre CXXV OpenAir - OpenAir Special Use Airspace Format

(GDAL/OGR >= 1.8.0)

Ce pilote lit les fichiers décrivant les *Special Use Airspaces* au format OpenAir.

Airspace sont retournés sous forme de features d'une couche unique appelée 'airspaces', avec une géométrie de type polygone et les champs suivants : CLASS, NAME, FLOOR, CEILING.

Les géométries Airspace composées d'arcs seront tessélées. Les informations de style lorsqu'elles sont présentes sont renvoyées au niveau de la feature.

Une couche supplémentaire appelée 'labels' contiendra une feature pour chaque label (élément AT). Il peut y avoir de multiple enregistrement AT pour un segment airspace AT. Les champs sont les mêmes que ceux de la couche 'airspaces'.

A Voir également

- Description du format OpenAir

Chapitre CXXVI PDS - Planetary Data Systems TABLE

(GDAL/OGR >= 1.8.0)

Ce pilote lit les objets TABLE d'un jeu de données PDS.

Notez qu'il y a un pilote PDS dans GDAL pour lire les objets IMAGE raster à partir de jeux de données PDS.

Un fichier label du produit doit être fourni au pilote (même quand les données sont placées dans un fichier séparé).

Si le fichier label contient un objet *TABLE*, il sera lu comme la seule couche du jeu de données.

Si aucun objet *TABLE* n'est trouvé, le pilote cherchera tous les objets contenant la chaîne TABLE et lire chacune d'elle dans une couche.

Les tables ASCII et BINARY sont gérées. Le pilote peut récupéré les descriptions de champs à partir d'un objet COLUMN en ligne ou à partir d'un fichier séparé pointé par ^STRUCTURE.

Si la table a des colonnes LONGITUDE et LATITUDE de type REAL et avec UNIT=DEGREE, elles seront utilisées pour renvoyer les géométries POINT.

A Voir également

- Description du format PDS (voir l'annexe A.29 du fichier StdRef_20090227_v3.8.pdf).

Chapitre CXXVII PostgreSQL / PostGIS

Le pilote implémente la gestion de l'accès aux tables spatiales dans PostgreSQL étendue par la gestion des données spatiales PostGIS. Une gestion existe dans le pilote pour utiliser PostgreSQL sans PostGIS mais avec moins de fonctionnalité.

Ce pilote nécessite une connexion à une base Postgres. Si vous voulez préparer un dump SQL pour l'injecter plus tard dans une base Postgres, vous pouvez utiliser plutôt :ref:`gdal.ogr.formats.pgdump` (GDAL/OGR >= 1.8.0).

Unknown interpreted text role "ref".

Vous pouvez trouver des informations additionnelles sur le pilote dans la page :ref:`gdal.ogr.formats.pg_advanced`.

Unknown interpreted text role "ref".

A Connexion à une base de données

Pour se connecter à une source de données Postgres, utilisez une chaîne de connections définissant le nom de la base de données, avec autant de paramètres supplémentaires que nécessaire :

```
PG:dbname=databasename
```

ou

```
PG:"dbname='databasename' host='addr' port='5432' user='x'
password='y'"
```

Il est également possible d'omettre le nom de la base de données et se connecter à celle par défaut, avec le même nom comme le nom d'utilisateur.

Note!

on utilise PQconnectdb() pour réaliser la connexion, n'importe quelles autres options et valeurs par défauts qui pourraient s'y appliquer, s'applique au nom (référez vous à la documentation du serveur PostgreSQL ici pour PostgreSQL 8.4). Le préfixe PG: est

utilisé pour marquer le nom en tant que chaîne de connexion postgres.

B Les colonnes géométriques

Si la table *geometry_columns* existe, alors toutes les tables listées et les vues nommées seront traitées comme des couches OGR. Autrement toutes les tables utilisateurs attributaires seront traitées comme des couches.

À partir de GDAL 1.7.0, le pilote gère également la colonne de type geography introduit dans PostGIS 1.5.

C Requêtes SQL

Le pilote PostgreSQL envoie les commandes SQL directement à PostgreSQL par défaut, plutôt que de les évaluer en interne lors de l'utilisation de l'appel *ExecuteSQL()* sur *OGRDataSource*, ou l'option en ligne de commande *-sql* pour `ogr2ogr`. Les expressions des requêtes attributaires sont également envoyées à PostgreSQL. Il est aussi possible de questionner le pilote pour prendre en charge les commandes avec le moteur :ref:`gdal.ogr.sql`, en passant la chaine *OGRSQL* à la méthode *ExecuteSQL()*, comme nom du dialecte SQL.

Unknown interpreted text role "ref".

Le pilote PostgreSQL dans OGR gère les appels *OGRDataSource::StartTrasaction()*, *OGRDataSource::CommitTransaction()* et *OGRDataSource::RollbackTransaction()* dans le sens normal de SQL.

D Problèmes lors de la création

Le pilote PostgreSQL ne gère pas la création de nouveau jeu de données (une base de données dans PostgreSQL), mais il permet la création de nouvelles couches dans une base de données existante.

Comme mentionné au-dessus, le système du type est pauvre, et plusieurs types d'OGR ne sont pas mappés correctement dans PostgreSQL.

Si la base de données a les types PostGIS chargés (c'est à dire le type *geometry*) les nouvelles couches crées seront crées avec le type géométrique de PostGIS. Autrement elles utiliseront un type OID. Par défaut il est supposé que le texte envoyé à Postgres est encodé au format UTF-8. Cela convient pour l'ASCII, mais peut entrainer des erreurs pour les caractères étendus (ASCII 155+ par exemple). Bien que OGR ne fournisse aucun contrôle direct pour cela, vous pouvez définir la variable d'environnement *PGCLIENTENCODING* pour indiquer le format qui est utilisé. Par exemple, si votre texte est LATIN1 vous pouvez définir la variable d'environnement à LATIN1 avant d'utiliser OGR et les données en entrées seront supposées être en LATIN1 au lieu de UTF-8.

Un moyen alternatif pour définir l'encodage du client est d'utiliser la commande SQL suivante avec *ExecuteSQL()* : "`SET client_encoding TO encoding_name`" où *encoding_name* est LATIN1, etc.

Les erreurs peuvent être récupérées en englobant cette commande avec la paire *CPLPushErrorHandler()/CPLPopErrorHandler()*.

1 Options de création de jeu de données

Aucune.

2 Options de création de couches

- **GEOM_TYPE :** l'option de création de couche *GEOM_TYPE* peut être définie à *Geometry*, "geography" (PostGIS >= 1.5), *BYTEA* ou *OID* pour forcer le type de la géométrie utilisée pour une table. Pour une base de données PostGIS, "geometry" est la valeur par défaut.
- **OVERWRITE :** Il peut être définie à *YES* pour forcer une couche existante du nom désiré à être détruite avant la création de la couche demandée.
- **LAUNDER :** Il peut être définie à *YES* pour forcer les nouveaux champs crées sur cette couche à avoir les noms de champs "nettoyer" dans une forme compatible avec PostgreSQL. Cela convertie en minuscule et convertie certains caractères spéciaux comme "-" et "#" en "_". Si *NO* les noms exacts seront préservés. La valeur par défaut est *YES*. Si activé le nom de la table (couche) sera également nettoyer.
- **PRECISION :** Il peut être définie à *YES* pour forcer la création de nouveaux champs dans cette couche pour essayer de représenter l'information de précision et longueur, si disponible en utilisant les types *NUMERIC(width,precision)* ou *CHAR(width)*. Si *NO* alors les types *FLOAT8*, *INTEGER* et *VARCHAR* seront utilisé à la place. *YES* pas défaut.
- **DIM={2,3} :** Contrôle la dimension de la couche. 3 par défaut. Important à définir à 2 pour les couches 2D avec PostGIS 1.0+ puisqu'il a des contraintes sur la dimension de la géométrie pendant le chargement.
- **GEOMETRY_NAME :** définie le nom de la colonne géométrique dans une nouvelle table. S'il est omis, sera définie par défaut à *wkb_geometry* pour GEOM_TYPE=geometry, ou *the_geog* pour GEOM_TYPE=geography..
- **SCHEMA :** Définie le nom du schéma pour une nouvelle table. L'utilisation d'un même nom de couche dans un schéma différent est gérée, mais pas dans un schéma public ou autres. Notez que l'utilisation de l'option *-overwrite* de `ogr2ogr` et de l'option *-lco SCHEMA=* en même temps ne fonctionnera pas, puisque la commande `ogr2ogr` ne comprendra pas que la couche existante doit être détruite dans le schéma défini. Utilisez l'option *-nln* de `ogr2ogr` à la place, ou mieux la chaîne de connexion *active_schema*. Voir ci-dessous les exemples.
- **SPATIAL_INDEX :** (à partir de GDAL 1.6.0) Définie à *ON* par défaut. Créer un index spatial sur la colonne géométrique pour accélérer les requêtes. Définissez-la à *OFF* pour la désactiver (a un effet seulement quand PostGIS est disponible).
- **TEMPORARY :** (à partir de GDAL 1.8.0) définie à OFF par défaut. créé une table temporaire au lieu d'une table permanente.
- **NONE_AS_UNKNOWN :** (à partir de GDAL 1.8.1) peut être définie à TRUE pour forcer les couches non-spatiales (wkbNone) à être créées comme table spatiale de type GEOMETRY (wkbUnknown), qui était le comportement avant GDAL 1.8.0. NO par défaut, auquel cas une table régulière est créée et non enregistré dans la table geometry_columns de PostGIS.
- **FID :** (à partir de GDAL 1.9.0) nom de la colonne FID à créer. 'ogc_fid' par défaut.
- **EXTRACT_SCHEMA_FROM_LAYER_NAME :** (à partir de GDAL 1.9.0) peut être définie à NO pour éviter de considérer le caractère "." comme séparateur entre le schéma et le nom de la table. YES par défaut.

3 Options de configuration

Il y a une variété d'options de configuration qui aide à contrôler le comportement de ce pilote.

- **PG_USE_COPY :** il peut être à "YES" pour utiliser *COPY* pour l'insertion de données dans PostgreSQL. ''COPY'' est moins robuste que *INSERT*, mais significativement plus rapide.
- **PGSQL_OGR_FID :** définie le nom d'une clé primaire au lieu de 'ogc_fid'. Utiliser seulement lors de l'ouverture d'une couche dont la clé primaire ne peut pas être autodétectée. Ignoré par *CreateLayer()* qui utilise l'option de création FID.

Bullet list ends without a blank line; unexpected unindent.

Explicit markup ends without a blank line; unexpected unindent.

- **PG_USE_BASE64 :** (GDAL >= 1.8.0) si définie à "YES", les géométries seront récupérées encodées en EWKB BASE64 au lieu de la forme canonique EWKB HEX. Cela réduit la quantité de données transférée de 2 N à 1.333 N, où N est la taille des données EWKB. Cependant, cela peut être un peu plus lent que récupérer la forme canonique quand le client et le serveur sont sur la même machine, la valeur par défaut est donc NO.

E Exemples

- Des traductions simples de shapefile dans PostgreSQL. la table 'abc' sera crée avec les géométries de *abc.shp* et les attributs de *abc.dbf*. L'instance de base de données (warmerda) doit déjà exister, et la table *abc* ne doit pas être crée.

```
% ogr2ogr -f PostgreSQL PG:dbname=warmerda abc.shp
```

- Ce second exemple charge une couche des limites des pays à partir d'un VPF (via le pilote OGDI), et renomme le nom énigmatique de la couche OGDI en un nom plus lisible. Si une table existante du nom désiré existe, elle sera écrasée.

```
% ogr2ogr -f PostgreSQL PG:dbname=warmerda \
          gltp:/vrf/usr4/mpp1/v0eur/vmaplv0/eurnasia \
          -lco OVERWRITE=yes -nln polbndl_bnd
'polbndl@bnd(*)_line'
```

- Dans cet exemple nous fusionnons des données lignes tiger de deux répertoires différents de fichier tiger dans une table. Notez que la seconde invocation utilise *-append* et pas *OVERWRITE=yes*.

```
% ogr2ogr -f PostgreSQL PG:dbname=warmerda tiger_michigan \
       -lco OVERWRITE=yes CompleteChain
% ogr2ogr -update -append -f PostgreSQL PG:dbname=warmerda
tiger_ohio \
       CompleteChain
```

- Cet exemple montre l'utilisation d'`ogrinfo` pour évaluer une commande de requête SQL dans PostgreSQL. Des requêtes PostGIS plus sophistiquées peuvent

être utilisées également via la commande -sql dans `ogrinfo`.

```
ogrinfo -ro PG:dbname=warmerda -sql "SELECT pop_1994 from canada
where province_name = 'Alberta'"
```

- Cet exemple montre l'utilisation de `ogrinfo` pour lister les couches PostgreSQL/PostGIS sur un hôte différent.

```
ogrinfo -ro PG:'host=myserver.velocet.ca user=postgres
dbname=warmerda'
```

1 FAQ

- **Pourquoi ne puis pas voir mes tables ? PostGIS est installé et j'ai des données.**

  Vous devez avoir les permissions sur toutes les tables que vous voulez lire *et* geometry_columns et spatial_ref_sys.
  Un comportement erroné peut ne renvoyer aucun message d'erreur si vous n'avez pas la permission à ces tables. Les problèmes de permission sur les tables *geometry_columns* et/ou *spatial_ref_sys* peut être généralement confirmés si vous pouvez voir les tables en définissant l'option de configuration *PG_LIST_ALL_TABLES* à YES. (par exemple `ogrinfo --config PG_LIST_ALL_TABLES YES PG:xxxxx`).

F Lisez également

- :ref:`gdal.ogr.formats.pg_advanced`
  Unknown interpreted text role "ref".
- :ref:`gdal.ogr.formats.pgdump`
  Unknown interpreted text role "ref".
- Page principale de PostgreSQL
- PostGIS
- PostGIS en Français
- Page d'exemples dans le wiki sur PostGIS / OGR

Chapitre CXXVIII Dump SQL pour PostgreSQL

(GDAL/OGR >= 1.8.0)

Duplicate explicit target name: "postgis".

Ce pilote en écriture seul implémente la gestion de la génération de fichier dump SQL qui peut être injecté plus tard dans une instance live de PostgreSQL. Il gère la version étendue de PostgreSQL avec les géométries PostGIS.

Ce pilote est très similaire à la commande shp2pgsql de PostGIS.

La plupart des options de création sont partagé avec le pilote normale PostgreSQL.

Le pilote ouvre le fichier en sortie via l'API VSIF Large, il est donc possible de définir /vsistdout/ comme fichier en sortie pour tout envoyer dans la sortie standard.

A Options de création

1 Options de création de jeu de données

- **LINEFORMAT :** par défaut les fichiers sont créés avec la convention de terminaison de ligne de la plateforme locale (CR/LF sous win32 ou LF sur tous les autres systèmes). Cela peut être écrasé par l'utilisation de l'option de création de couche LINEFORMAT qui peut avoir les valeurs **CRLF** (format DOS) ou **LF** (format Unix).

2 Options de création de couche

- **GEOM_TYPE :** l'option de création de couche *GEOM_TYPE* peut être définie à *Geometry*, "geography" (PostGIS >= 1.5), pour forcer le type de la géométrie utilisée pour une table. "geometry" est la valeur par défaut.
- **LAUNDER :** Il peut être définie à *YES* pour forcer les nouveaux champs crées sur cette couche à avoir les noms de champs "nettoyer" dans une forme compatible avec PostgreSQL. Cela convertie en minuscule et convertie certains caractères spéciaux comme "-" et "#" en "_". Si *NO* les noms exacts seront préservés. La valeur par défaut est *YES*. Si activé le nom de la table (couche) sera également

nettoyer.

- **PRECISION :** Il peut être définie à *YES* pour forcer la création de nouveaux champs dans cette couche pour essayer de représenter l'information de précision et longueur, si disponible en utilisant les types *NUMERIC(width,precision)* ou *CHAR(width)*. Si *NO* alors les types *FLOAT8*, *INTEGER* et *VARCHAR* seront utilisé à la place. *YES* pas défaut.
- **DIM={2,3} :** Contrôle la dimension de la couche. 3 par défaut. Important à définir à 2 pour les couches 2D avec PostGIS 1.0+ puisqu'il a des contraintes sur la dimension de la géométrie pendant le chargement.
- **GEOMETRY_NAME :** définie le nom de la colonne géométrique dans une nouvelle table. S'il est omis, sera définie par défaut à *wkb_geometry* pour GEOM_TYPE=geometry, ou *the_geog* pour GEOM_TYPE=geography..
- **SCHEMA :** Définie le nom du schéma pour une nouvelle table. L'utilisation d'un même nom de couche dans un schéma différent est gérée, mais pas dans un schéma public ou autres.
- **CREATE_SCHEMA :** (OGR >= 1.8.1) à utiliser en combinaison avec SCHEMA. Définie à ON par défaut afin que l'instruction CREATE SCHEMA soit émise. Placez à OFF pour empêcher CREATE SCHEMA d'être envoyée.
- **SPATIAL_INDEX :** définie à *ON* par défaut. Créé un index spatial sur la colonne géométrique pour accélérer les requêtes. Définissez-la à *OFF* pour la désactiver (a un effet seulement quand PostGIS est disponible).
- **TEMPORARY :** définie à OFF par défaut. Créé une table temporaire au lieu d'une table permanente.
- **WRITE_EWKT_GEOM :** définie à OFF par défaut. Placez à ON pour écrire des géométrie EWKT au lieu de géométries HEX. Cette option n'aura pas d'effet si la variable d'environnement PG_USE_COPY est à YES.
- **CREATE_TABLE :** définie à ON par défaut afin que les tables soient recréées si nécessaire. Placer à OFF pour désactiver cela et utiliser une structure de table existante.
- **DROP_TABLE :** (OGR >= 1.8.1) définie à ON par défaut afin que les tables soient détruites avant d'être recréées. Placer à OFF pour éviter que `DROP TABLE` ne soit envoyé. Définissez le en IF_EXISTS dans le but d'émettre `DROP TABLE IF EXISTS` (nécessite PostgreSQL >= 8.2).
- **SRID :** définie le SRID de la géométrie. -1 par défaut sauf si un SRS est associé avec la couche. Dans ce cas, si le code EPSG est mentionné, il sera utilisé comme SRID (Notez que la table spatial_ref_sys doit être correctement remplie avec le SRID définie).

3 Variables d'environnement

- **PG_USE_COPY :** peut être à "YES" pour utiliser COPY lors de l'insertion des données dans Postgresql. COPY est moins robuste que INSERT, mais est significativement plus rapide.

4 Exemple

- translation simple d'un shapefile vers PostgreSQL dans un fichier abc.sql. la table 'abc' sera créée avec les features à partir de abc.shp et les données attributaires à partir de abc.dbf. Le SRID est spécifié. PG_USE_COPY est définie à YES pour améliorer les performances.

```
% ogr2ogr --config PG_USE_COPY YES -f PGDump abc.sql abc.shp
-lco SRID=32631
```

- renvoyer la sortie du pilote PGDump vers la commande psql.

```
% ogr2ogr --config PG_USE_COPY YES -f PGDump /vsistdout/ abc.shp
| psql -d my_dbname -f -
```

5 Voir également

- :ref:`gdal.ogr.formats.pg`
  Unknown interpreted text role "ref".
- Page principale de PostgreSQL
- Duplicate explicit target name: "postgis".
  PostGIS
- Page d'exemple sur PostGIS / OGR Wiki

Chapitre CXXIX ESRI Personal GeoDatabase

OGR gère en option la lecture de fichier .mdb des Géodatabases Personnelles via ODBC. Les Géodatabases Personnelles sont une base de données Microsoft Access(r) avec un ensemble de tables définies par ESRI pour prendre en compte les méta-données de la géodatabase, et avec des géométries pour les objets contenu dans une colonne BLOB dans un format personnalisé (essentiellement des fragments de géométries de Shapefile). Ces pilotes accèdent aux géodatabases personnelles via ODBC, mais ne dépend d'aucuns middleware d'ESRI.

Les Géodatabases Personnelles sont accédées en passant le nom du fichier .mdb pour être accédé comme source de données. Sous Windows, aucun DSN d'ODBC n'est nécessaire. Sous Linux, il y a un problème avec des connexions sans DSN dû à une implémentation incomplète ou bugguée de cette fonctionnalité dans le paquet MDB Tools. Il est donc nécessaire de configurer un Data Source Name (DSN) si le pilote MDB Tools est utilisé (vérifiez les instructions ci-dessous).

Dans le but de faciliter la compatibilité avec différentes configurations, l'option de configuration PGEO_DRIVER_TEMPLATE a été ajouté pour fournir une façon pour définir par programmation le DSN avec le nom du fichier comme argument. Dans le cas où le nom du pilote est connu, cela permet la construction de DSN basé sur cette information d'une manière identique à celle par défaut (utilisé pour Windows access pour le pilote Microsoft Access).

OGR traite toutes les tables de géométrie comme des couches. La plupart des géométries devraient être gérée, données 3D incluses. Les informations de mesure seront ignorées. Les informations du système de coordonnées devraient être proprement associées aux couches.

Pour l'instant, le pilote de Géodatabase Personnelle d'OGR ne se sert pas des index spatiaux pour accélérer les requêtes spatiales, bien que cela puisse être ajouté dans le futur.

Par défaut, les commandes SQL sont envoyées directement au moteur de base de données MDB. Il est également possible de demander au pilote de prendre en charge les commandes SQL avec le moteur :ref:`gdal.ogr.sql`, en définissant la chaine *OGRSQL* à la méthode *ExecuteSQL()*, comme nom du dialecte SQL.

Unknown interpreted text role "ref".

A Comment utiliser le pilote PGeo avec unixODBC et MDB Tools

À partir de GDAL/OGR 1.9.0, le pilote :ref:`gdal.ogr.formats.mdb` est une autre façon de lire les fichiers .mdb géodatabase personnelle d'ESRI sans nécessité unixODBC et les outils MDB.

Unknown interpreted text role "ref".

Cet article donne les explications étape par étape pour utiliser OGR avec le paquet unixODBC et pour accéder aux Géodatabases Personnelles avec le pilote PGeo.

1 Pré-requis

- Installez unixODBC >= 2.2.11
- Installez MDB Tools >= 0.6. La version 0.5.99 a également été testée par le développeur (pré-version 0.6).

(sur Ubuntu 8.04 : `sudo apt-get install unixodbc libmdbodbc`)

2 Configuration

Il y a deux fichiers de configuration pour unixODBC :

- *odbcinst.ini* - ce fichier contient la définition des pilotes ODBC disponibles à tous les utilisateurs ; ce fichier peut être trouvé dans le répertoire */etc* ou la localisation donnée par */--sysconfdir* SI VOUS COMPILEZ `unixODBC` vous-même.
- *odbc.ini* - ce fichier contient la définition des sources de données ODBC (entré DSN) disponibles à tous les utilisateurs.
- *~/.odbc.ini* - le fichier privé où les utilisateurs peuvent placer leurs propres sources de données ODBC.

Le format des fichiers de configuration est très simple :

```
[section_name]
entry1 = value
entry2 = value
```

Pour plus de détails, référez-vous au manuel d'unixODBC.

2.a 1. Configuration du pilote ODBC

D'abord , vous devez configurer le pilote ODBC pour accéder aux bases de données Microsoft Access(r) avec MDB Tools. Ajoutez la définition suivante à votre fichier *odbcinst.ini*.

```
[Microsoft Access Driver (*.mdb)]
Description = MDB Tools ODBC drivers
Driver     = /usr/lib/libmdbodbc.so.0
Setup      =
FileUsage  = 1
CPTimeout  =
CPReuse    =
```

- *[Microsoft Access Driver (*.mdb)]* - souvenez-vous d'utiliser "Microsoft Access Driver (*.mdb)*" *comme nom de la section parce que le pilote PGeo créé la chaine de connections ODBC pour les Géodatabases Personnelles en utilisant la chaine "DRIVER=Microsoft Access Driver (*.mdb);".
- *Description* - courte description de cette définition du pilote.
- *Driver* - chemin complet de pilote ODBC pour MDB Tools.

2.b 2. Configuration de la source de données ODBC

Dans cette section, on utilise 'sample.mdb' comme nom de la géodatabase personnelle, remplacez la avec le nom de votre propre base de données.

Créez un fichier *.odbc.ini* dans votre répertoire *HOME* :

```
$ touch ~/.odbc.ini
```

Placez-y la définition de la source de données ODBC suivante dans ce fichier :

```
[sample_pgeo]
Description = Sample PGeo Database
Driver      = Microsoft Access Driver (*.mdb)
Database    = /home/mloskot/data/sample.mdb
Host        = localhost
Port        = 1360
User        = mloskot
Password    =
Trace       = Yes
TraceFile   = /home/mloskot/odbc.log
```

Explication étape par étape d'une entrée DSN :

- *[sample_pgeo]* - c'est le nom de la source de données ODBC (DSN). Vous vous référez à votre Géodatabase personnelle en utilisant ce nom. Vous pouvez utiliser votre propre nom ici.
- *Description* - courte description de l'entrée DSN.
- *Driver* - nom complet du pilote défini à l'étape 1, au-dessus.
- *Database* - chemin complet vers le fichier *.mdb* de votre Géodatabase Personnelle.
- entré *Host*, *Port*, *User* et *Password* ne sont pas utilisé par le pilote MDB Tools.

3 Tester le pilote PGeo avec ogrinfo

Maintenant vous pouvez tester l'accès à une source de données PGeo avec `ogrinfo`.

D'abord, vérifiez que vous avez le pilote PGeo compilé dans OGR :

```
$ ogrinfo --formats
Supported Formats:
    ESRI Shapefile
    ...
    PGeo
    ...
```

Maintenant vous pouvez accéder à votre Géodatabase personnelle. Comme source de données utilisez *PGeo:<DSN>* où *<DSN>* est un nom d'une entré DSN que vous avez

placé dans votre fichier *.odbc.ini*.

```
ogrinfo PGeo:sample_pgeo
INFO: Open of `PGeo:sample_pgeo'
using driver `PGeo' successful.
1. ...
```

Après avoir lancé la commande ci-dessous, vous devez obtenir la liste des couches stockée dans votre géodatabase.

Maintenant, vous pouvez réaliser une requête pour détailler une couche particulière :

```
ogrinfo PGeo:sample_pgeo <layer name>
INFO: Open of `PGeo:sample_pgeo'
using driver `PGeo' successful.

Layer name: ...
```

B Ressources

- À propos des geodatabase d'ESRI
- [mdbtools-dev] les connexions sans DSN non géré ?

C Voir également

- :ref:`gdal.ogr.formats.mdb`
  Unknown interpreted text role "ref".

Chapitre CXXX IHO S-57 (ENC)

Warning!

Attention : *La traduction de cette page n'est pas parfaite. La traduction du terme sounding en sondage n'est peut être pas parfaite. N'hésitez pas à contacter l'auteur de cette page ou à l'éditer. Merci.*

Les jeux de données S-57 d'IHO sont gérés en lecture seule.

Le module pilote du format S-57 produit des objets pour tous les objets S-57 dans un fichier S-57 et les mises à jour associées. Les fichiers S-57 (ENC) ont normalement l'extension ".000".

Les objets *features* du format S-57 sont traduit en *features*. Les objets géométriques sont automatiquement collectés et formés dans les géométries sur l'objet.

Le lecteur du format S-57 est fonction de la présence de deux fichiers de gestion, * s57objectclasses.csv* et *s57attributes.csv* lors de son fonctionnement dans le but de traduire les objets en une classe d'objet d'une manière spécifique. Ceux-ci doivent être dans le répertoire pointé par la variable d'environnement *S57_CSV* ou dans le répertoire de travail actuel.

Les fichiers de mise à jour du format S-57 contiennent des informations sur la manière de mettre à jour un fichier de données S-57 distribué. Les fichiers de base ont normalement l'extension .000 tandis que les fichiers de mis à jour ont une extension comme .001, .002 etc. Le lecteur S-57 lira et appliquera normalement tous les fichiers de mis à jour à la volé de la version en mémoire du fichier de base. Les données des objets fournit à l'application inclus alors toutes les mises à jours.

A Traduction d'objet

Normalement tous les objets lu d'un fichier S-57 sont assigné à une couche basée sur le nom de la classe d'objet (*OBJL*) à laquelle ils appartiennent. Par exemple, avec un *OBJL* de valeur 2, l'objet est classé dans "*Airport / airfield*" et possède un nom court "*AIRARE*" qui est utilisé comme nom de couche. Un transfert typique d'un fichier S-57 aura plus de 100 couches.

Chaque type d'objet a un ensemble prédéfinie d'attributs comme définie par le standard

S-57. Par exemple, la classe d'objet aéroport (*AIRARE*)peut avoir les attributs *AIRARE*, *CATAIR*, *CONDTN*, *CONVIS*, *NOBJNM*, *OBJNAM*, *STATUS*, *INFORM*, *NINFOM*, *NTXTDS*, *PICREP*, *SCAMAX*, *SCAMIN*, *TXTDSC*, *RECDAT*, *RECIND*, *SORDAT*, et *SORIND*. Ces noms courts peuvent être liés à des noms plus long, et plus parlant en utilisant un catalogue d'attribut/objet S-57 comme le standard S-57 lui-même , ou le fichier catalogue (*s57attributes.csv*, et *s57objectclasses.csv*). Un tel cataloque peut également être utilisé pour établir toutes les classes d'objet disponible, et leurs attributs.

Ce qui suit sont des attributs communs, incluant des attributs génériques qui apparaissent sur tous les objets, sans regard de l'activation de la classe de l'objet :

Nom de l'attribut	***Description***	***Définie pour***
GRUP	Nombre du groupe	Tous les objets
OBJL	Code de l'étiquette de l'objet. Ce nombre indique la classe de l'objet de l'objet.	Tous les objets
RVER	Version de l'enregistrement	
AGEN	Code d'agence numérique, tel que *50* pour le *Canadian Hydrographic Service*. Une liste potentiellement non à jour est disponible dans le fichier *agencode.txt*.	Tous les objets
FIDN	Nombre d'identification de l'objet.	Tous les objets
FIDS	Subdivision d'identification de l'objet.	Tous les objets
DSNM	Nom du jeu de données. Le nom du fichier d'où provient l'objet. Utilisé avec l'attribut *LNAM* pour former un unique identifiant de gros jeu de données pour un objet.	Tous les objets
INFORM	Texte d'information	Certains objets
NINFOM	Texte d'information dans la langue nationale	Certains objets
OBJNAM	Nom de l'objet	Certains objets
NOBJNM	Nom de l'objet dans la langue nationale	Certains objets
SCAMAX	Échelle maximale pour l'affichage	Certains objets
SCAMIN	Échelle minimale pour l'affichage	Certains objets
SORDAT	Date de la source	Certains objets

Ce qui suit est présent si *LNAM_REFS* est activé :

Nom de l'attribut	***Description***	***Définie pour***
LNAM	Nom long. Un encodage des attributs *AGEN*, *FIDN* et *FIDS* utilisés pour identifier de manière unique cet objet dans un fichier S-57.	Tous les objets
LNAM_REFS	Liste de noms long d'objets liés	Tous les objets

FFPT_RIND	Indicateurs de relation pour chaque relation *LNAM_REFS*	Tous les objets

B Sondages

La profondeur du sondage est pris en charge d'une manière assez spéciale dans le format S-57, dans l'objectif de représenter efficacement les nombreux points de données disponibles. Dans le format S-57 un objet sondage peut avoir plusieurs points de sondages. Le lecteur S-57 découpe chacun d'eux en son propre type d'objet *SOUNDG* d'objet avec un *s57_type* de '*s57_point3d*'. Tous les sondages d'un enregistrement d'objet isolé possédera la même valeur *AGEN*, *FIDN*, *FIDS* et *LNAM*.

C Options de contrôle du format S-57

Il y a plusieurs options de contrôle qui peuvent être utilisées pour modifier le comportement du lecteur S-57. Les utilisateurs peuvent définir ceux-ci en les ajoutant dans la variable d'environnement *OGR_S57_OPTIONS*.

- **UPDATES=APPLY/IGNORE :** est ce que les fichiers mis à jour doivent être incorporés dans les données de base à la volée.. *APPLY* par défaut.
- **SPLIT_MULITPOINT=ON/OFF :** est ce que les les sondages multipoint doivent être découpées en plusieurs objets sondage. Les géométries multipoint ne sont pas bien pris en charge par de nombreux formats, il peut donc être intéressant de découpe les objets sondage avec de nombreux points en plusieurs objets points seuls. *OFF* par défaut.
- **ADD_SOUNDG_DEPTH=ON/OFF :** est ce qu'une attribut *DEPTH* doit être ajouté aux objets *SOUNDG* et y assigner la profondeur du sondage. Cela doit seulement être activé lorsque *SPLIT_MULTIPOINT* est aussi activé. *OFF* par défaut.
- **RETURN_PRIMITIVES=ON/OFF :** est ce que toutes les primitives géométriques de bas niveau doivent être retournées comme IsolatedNode, ConnectedNode, Edge et couche Face. *OFF* par défaut.
- **PRESERVE_EMPTY_NUMBERS=ON/OFF :** si activé, les attributs numériques assignant une chaîne vide comme valeur seront préservés comme une valeur numérique spéciale. Cette option ne doit généralement pas être utilisée, mais peut être utile lors d'une translation du format S-57 vers le format S-57 sans perte. *OFF* par défaut.
- **LNAM_REFS=ON/OFF :** est ce que les champs *LNAM* et *LNAM_REFS* doivent être attachés aux objets reflétant la relation d'objet à objet dans le groupe FFPT du fichier S-57. *OFF* par défaut.
- **RETURN_LINKAGES=ON/OFF :** est ce que des attributs additionnels lié aux objets doivent être attachés à leur géométrie sous-jacente. Ce sont les valeurs du groupe FSPT, et sont d'abord nécessaire lors d'une translation du format S-57 vers le format S-57. *OFF* par défaut.

Exemple :

```
set OGR_S57_OPTIONS =
"RETURN_PRIMITIVES=ON,RETURN_LINKAGES=ON,LNAM_REFS=ON"
```

D Export du formar S-57

La possibilité d'export du S-57 préliminaire a été ajoutée à la version 1.2.0 de GDAL/OGR mais a seulement pour une utilisation spécialisée, et n'est pas documenté à ce jour. Définir les options suivantes est le minimum nécessaire pour gérer la conversion du format S-57 vers le format S-57 via OGR.

```
set OGR_S57_OPTIONS =
"RETURN_PRIMITIVES=ON,RETURN_LINKAGES=ON,LNAM_REFS=ON"
```

E Voir également

- Catalogue des objets/attributs en ligne de S-57
- Page de S-57 de Frank : liens vers d'autres ressources et des jeux de données échantillons.

Chapitre CXXXI ESRI ArcSDE

OGR gère en option la lecture des instances de bases de données ArcSDE d'ESRI. ArcSDE est une solution spatiale de type middleware pour le stockage de données spatiales pour diverses bases de données relationnelles en arrière. Le pilote ArcSDE d'OGR dépend de sa compilation avec les bibliothèques cliente d'ArcSDE fournit par ESRI.

Les instances ArcSDE sont accédées avec un nom de source de données de la forme suivante. Les champs serveur, instance, nom d'utilisateur et mot de passe sont nécessaire. L'instance est le numéro de port du serveur SDE, qui est par défaut à 5151. Si le paramètre couche est définie alors le pilote SDE est capable de sauter la lecture des méta-données pour chaque couche ; échapper cette étape est un moyen d'aller plus vite.

Note!

Seulement GDAL 1.6+ gère la requête d'opération d'écriture et de version. Les versions plus anciennes ne gère que les requêtes sur la version de la base (SDE.DEFAULT) et aucune opération d'écriture.

```
SDE:server,instance,database,username,password[,layer]
```

Pour définir une version sur laquelle réaliser une requête , vous **devez** définir une couche. La version *SDE.DEFAULT* sera utilisé lorsqu'aucune nom de version n'est définie.

```
SDE:server,instance,database,username,password,layer,[version]
```

Vous pouvez également faire une requête pour créer une nouvelle version si elle n'existe pas déjà. Si la version enfant existe déjà, elle sera utilisé à moins que la variable d'environnement *SDE_VERSIONOVERWRITE* est définie à *TRUE*. Dans ce cas, la version sera supprimée et recrée.

```
SDE:server,instance,database,username,password,layer,[parentversion],
[childversion]
```

Le pilote ArcSDE d'OGR ne gère pas la lecture des données CAD (traité comme attribut BLOB), les propriétés annotation, les valeurs de mesure au sommet, ou les données raster. La méthode *ExecuteSQL()* n'est pas passé à la base de données sous-jacente. Pour l'instant il est interprété par le parseur SQL limité d'OGR. Les indexes spatiaux sont utilisés pour accélérer les requêtes spatiales.

Le pilote a été testé avec ArcSDE 9.x, et devrait fonctionner avec les versions plus récentes, ainsi que les version ArcSDE 8.2 ou 8.3. À la fois les géométries 2D et 3D sont gérées. Les géométries Courbe sont approximées comme des lignes (en réalité encore à faire).

ArcSDE est généralement sensible à la casse, et les noms de tables entièrement qualifiés. Bien que vous pouvez utiliser les noms courts pour certaines opérations, d'autres (notamment la suppression) nécessitera un nom entièrement qualifié. À cause de cela, il est généralement préférable de toujours utiliser des noms entièrement qualifiés.

A Options de création de couches

- **OVERWRITE :** peut être définie pour permettre l'écrasement d'une couche existante pendant le processus de création de couche. Si définie, et que la valeur n'est pas *NO*, la couche sera d'abord effacée avant d'être créé avec le même nom que celle existante. Définie à *NO* explicitement, ou ne pas inclure l'option pour traiter les tentatives de créer de nouvelles couches qui entrent en collision avec une couche existante de même nom comme une erreur. *Off* par défaut.
- **GEOMETRY_NAME :** par défaut OGR crée de nouvelles couches avec la colonne géométrique (*feature*) nommé 'SHAPE'. si vous désirez utiliser un nom différent, celui-ci peut être fournit par l'option de création d couche *GEOMETRY_NAME*.
- **SDE_FID :** peut être définie pour écraser le nom par défaut de la colonne ID des objets. *OBJECTID* par défaut.
- **SDE_KEYWORD :** le mot-clé *DBTUNE* avec lequel créer la couche. Par défaut à *DEFAULTS*.
- **SDE_DESCRIPTION :** La description textuelle de la couche. Par défaut à <<Created by GDAL/OGR 1.6>> (également utilisé comme la description de la version lors de la création d'une nouvelle création de version enfant à partir d'une version parent).
- **SDE_MULTIVERSION :** si cette option de création est définie à *FALSE*, le multi-versioning sera désactivé pour la couche au moment de la création. Par défaut, les tables multi-version sont créées quand des couches sont créées dans une source de données SDE.
- **USE_NSTRING :** si cette option est définie à "TRUE" alors les champs chaînes seront créé sous le type NSTRING. Cette option a été ajouté pour GDAL/OGR 1.9.0.

B Variables d'environnement

- **OGR_SDE_GETLAYERTYPE :** peut être à *TRUE* pour déterminer le type de géométrie à partir de la base de données. Autrement, le pilote SDE retournera toujours un type de géométrie *Unknown*.
- **OGR_SDE_SEARCHORDER :** peut être à *ATTRIBUTE_FIRST* pour informer ArcSDE de filtrer sur un attribut **avant** d'utiliser un filtre spatial ou *SPATIAL_FIRST* pour utiliser un filtre spatial. Par défaut, il utilise un filtre spatial d'abord.
- **SDE_VERSIONOVERWRITE :** si définie à *TRUE*, la version enfant définie sera

supprimée avant d'être recréée. Notez que cette action ne fait rien pour concilier n'importe quelle édition qui existait sur cette version avant de faire cela et les rejette essentiellement.

- **GR_SDE_USE_NSTRING :** si cette option est définie à "TRUE" alors les champs chaînes seront créé sous le type NSTRING. Cette option a été ajouté pour GDAL/OGR 1.9.0.

C Exemples

Voyez le script test `ogr_sde.py` pour des exemples de chaîne de connections et des usages de ce pilote.

Chapitre CXXXII SDTS

Les jeux de données SDTS TVP (profile vectoriel topologique *Topological Vector Profile*) et *Point Profile* sont gérés en lecture. Chaque module attribut primaire, noeud (point), ligne et polygone sont créé comme une couche distincte.

Pour sélectionner un transfert SDTS, le nom du fichier catalogue doit être utilisé. Par exemple *TR01CATD.DDF* où les quatre premiers caractères sont ce qui typiquement varie.

Les informations des systèmes de coordonnées de SDTS sont proprement gérées pour la plupart des systèmes de coordonnées définie dans SDTS.

Il n'y a pas de gestion de création ou de mise à jour dans le pilote SDTS.

Notez que dans les jeux de données TVP la géométrie polygonal est formée à partir de la géométrie dans les modules lignes. Les attributs des modules d'attribut primaire doivent être proprement attachés à leur objet noeud, ligne ou polygone attaché, mais peuvent être accédé séparément de leur propre couche.

Ce pilote ne gère pas les jeux de données SDTS raster (DEM).

A Voir également

- Bibliothèque d'abstraction de SDTS : la bibliothèque de base utilisé pour implémenter le pilote.
- http://mcmcweb.er.usgs.gov/sdts : Page web sur principale SDTS de l'USGS.

Chapitre CXXXIII ESRI Shapefile

Toutes les variétés de Shapefiles d'ESRI devraient être disponibles en lecture et les fichiers 3D simple peuvent être créés.

Normalement le pilote OGR Shapefile traite un ensemble de répertoire contenant des shapefiles comme source de données, et un shapefile dans ce répertoire comme couche. Dans ce cas le nom du répertoire doit être utilisé comme nom de la source de données. Cependant, il est également possible un seul des fichiers (.shp, .shx or .dbf) dans le jeu du shapefile comme nom du jeu de données, et il sera alors traité comme un jeu de données avec une couche;

Notez que lors de la lecture d'un Shapefile de type *SHPT_ARC*, la couche correspondante sera reportée comme de type *wkbLineString* mais en fonction du nombre de parts de chaque géométrie, le type réel de la géométrie pour chaque objet peut être soit *OGRLineString* soit *OGRMultiLineString*. La même chose s'applique pour les Shapefiles *SHPT_POLYGON*, reporté comme couches de type *wkbPolygon* mais en fonction du nombre de parts de chaque géométrie, le type réel peut être soit *OGRPolygon* ou *OGRMultiPolygon*.

Les nouvelles valeurs de mesures d'ESRI seront ignorées si rencontrées. Les fichiers multipatch sont lu et chaque patch de la géométrie est retourné dans une représentation multipolygone avec un polygone par triangle dans les mailles et les fans du triangle.

Si des fichiers .prj dans l'ancien style Arc/Info ou le nouveau style ESRI en WKT de l'OGC sont présent, ils seront lus et utilisés pour associer une projection avec les géométries.

Le pilote de lecture suppose que les polygones multipart suivent la spécification, c'est-à-dire que les sommets des anneaux externes doivent être orientés dans le sens des aiguilles d'une montre sur le plan x/y, et ceux de l'anneau intérieur en sens inverse. Si un Shapefile est cassé, il est possible de définir l'option de configuration *OGR_ORGANIZE_POLYGONS=DEFAULT* afin de procéder à une analyse complète basée sur les relations topologiques des parties des polygones afin que les polygones résultants soient correctement définie selon la convention OGC Simple Feature.

Une tentative est faite pour lire la définition du LDID/codepage à partir du fichier .dbf et l'utiliser pour traduire les champs de chaîne en UTF-8 lors de la lecture, puis vers l'écriture. LDID "87 / 0x57" est considéré comme ISO8859_1 qui peut ne pas être approprié.

L'option de configuration SHAPE_ENCODING peut être utilisé pour écraser l'interprétation de l'encodage du shapefile par n'importe quel encodage géré par CPLRecode ou par "" pour éviter n'importe quel enregistrement (la gestion de l'enregistrement est nouveau dans GDAL/OGR 1.9.0).

A Indexe spatiale et attributaire

Le pilote OGR shapefile gère l'indexage spatial et une forme limitée d'indexage attributaire.

L'indexage spatial utilise les mêmes fichiers d'index spatial *quadtree* .qix qui sont utilisés par MapServer. !il ne gère pas les fichiers d'inde spatiale d'ESRI (.sbn / .sbx). L'index spatial peut accélérer les parsages avec des filtres spatiaux pour d'importants jeux de données pour récupérer une portion d'une zone rapidement.

Pour créer un index spatial, envoyer une commande SQL de la forme :

```
CREATE SPATIAL INDEX ON nomTable [DEPTH N]
```

où le paramètre optionnel ''DEPTH'' peut être utilisé pour contrôler le nombre de niveaux de l'arbre d'index généré. Si ''DEPTH'' est omis, la profondeur de l'arbre est estimée sur la base de nombre de géométrie dans un shapefile et son domaine de valeur de 1 à 12.

Pour effacer un index spatial, envoyez une commande de la forme :

```
DROP SPATIAL INDEX ON nomTable
```

Duplicate explicit target name: "mapserver".

Autrement, la commande shptree de MapServer peut être utilisée.

```
shptree <shpfile> [<depth>] [<index_format>]
```

Plus d'information est disponible sur cette commande sur la page shptree de MapServer.

Pour l'instant le pilote OGR shapefile gère seulement les indexes d'attribut pour rechercher des valeurs spécifiques dans une colonne de clé unique. Pour créer un index d'attribut pour une colonne, envoyez une commande SQL de la forme *CREATE INDEX ON nomTable USING nomchamp*. Pour supprimer des index d'attribut envoyez une commande de la forme *DROP INDEX ON nomTable*. L'index d'attribut accélèrera les recherches des clauses *WHERE* de la forme *nomChamp = valeur*. L'index d'attribut est en réalité stocké comme un index au format mapinfo et n'est pas compatible avec toute autre application shapefile.

B Problèmes de création

Le pilote shapefile traite un répertoire comme un jeu de données, et chaque ensemble Shapefile (.shp, .shx, and .dbf) comme une couche. Le nom du jeu de données sera créé comme un nom de répertoire. Si le répertoire existe déjà il est utilisé et les fichiers existants dans le répertoire sont ignorés. Si le répertoire n'existe pas, celui-ci sera créé.

Une tentative de création d'un nouveau jeu de données avec une extension .shp résultera en un ensemble de fichiers crée à la place d'un répertoire.

Les shapefile d'ESRI peuvent seulement stocker un type de géométrie par couche

(shapefile). En création cela peut être basé sur le fichier source (si un type de géométrie uniforme est connu par le pilote source), ou il peut être défini directement par l'utilisateur avec l'option de création SHPT (indiqué plus bas). S'il n'est pas connu, la création de la couche échouera. Si des géométries d'un type incompatible sont écrites vers la couche, l'envoi en sortie échouera avec un message d'erreur.

Notez qu'il peut rendre plus difficile la translation d'une couche de géométrie mixte d'un format vers un format Shapefile en utilisant `ogr2ogr`, puisque `ogr2ogr` ne gère pas la séparation des géométries d'une couche source. Lisez la FAQ pour une solution.

Les attributs des géométries du Shapefile sont stockés dans un fichier .dbf associé, et donc les attributs souffrent d'un certain nombre de limitations :

- les noms des attributs ne peuvent avoir qu'au maximum 10 caractères. Les noms plus longs seront tronqués. Cela peut entraîner des noms de colonnes qui ne sont pas uniques, qui sans aucun doute poseront problème plus tard.
- À partir de la version 1.7, le pilote Shapefile d'OGR tente de générer des noms de champs uniques. Des noms de champs dupliqués successifs, incluant ceux créés par troncation à 10 caractères, seront tronqué à 8 caractères et un numéro ajouté de 1 à 99.
  Par exemple :
  - a -> a, a -> a_1, A -> A_2;
  - abcdefghijk -> abcdefghij, abcdefghijkl -> abcdefgh_1
- Seul des champs de type *entier*, *réel* et *chaine de caractère* sont gérés (pas DateTime, juste year/month/day). Les champs de types *liste diverse* et *binaire* ne peuvent pas être crée.
- La largeur du champ et la précision sont directement utilisées pour établir la taille de stockage dans le fichier .dbf. Cela signifie que les chaines plus longues que la largeur du champ, ou les nombres qui ne remplissent pas les conditions du format du champ seront tronquées.
- les champs d'*entier* sans une largeur explicite sont traité avec une largeur de 11.
- les champs *réel* (point flottant) sans une largeur explicite sont traité avec une largeur de 24 avec 15 chiffres pour les décimales.
- les champs *caractères* sans une largeur assignée sont traité avec une largeur de 80 caractères.

Également, les fichiers .dbf doivent avoir au moins un champ. Si aucun n'est créé par l'application, un champ "FID" sera automatiquement créé et remplit des numéros d'enregistrement.

Le pilote shapefile d'OGR gère la réécriture des shapes existant dans un shapefile ainsi que la suppression des shapes. La suppression des shapes est notée comme supprimé dans le fichier .dbf, et seront ainsi ignoré par OGR. Pour les supprimer réellement (entrainant une renumérotation des FID) appeler la fonction SQL 'REPACK' par la méthode *ExecuteSQL()* de la source de données.

C Étendue spatiale

Shapefiles stocke l'étendue spatiale de al couche dans le fichier .SHP. L'étendue spatiale de la couche est automatiquement mis à jour lors de l'insertion d'une nouvelle feature dans le shapefile. Cependant lors de la mise à jour d'une feature existante, si la forme précédente a touché la bounding box de l'étendue spatiale mais la forme mise à jour ne touche pas le nouvelle étendue, l'étendue calculée ne sera pas correcte. Il sera alors nécessaire de forcer le calcul en invoquant la commande SQL 'RECOMPUTE EXTENT ON <tablename>' via la méthode *ExecuteSQL()* de la source de données. La même chose

s'applique pour la suppression d'une shape.

Note!

RECOMPUTE EXTENT ON est disponible à partir d'OGR >= 1.9.0.

D Problèmes sur les tailles

- Geometry : le format Shapefile utilise explicitement des offsets de 32 bits et ne put donc pas dépasser 8 Go (il utilise en réalité des offsets de 32 bit vers des mots de 16 bits). Par conséquent il n'est pas recommendé d'utiliser un fichier de plus de 4 Go.
- Attributs : le format dbf ne contient aucun offsets, il peut donc être arbitrairement large.

E Options de création de jeu de données

Aucune option.

F Options de création de couches

- **SHPT=type :** écrase le type de shapefile crée. Peut être une parmi NULL pour un simple fichier .dbf. avec aucun fichier .shp, *POINT*, *ARC*, *POLYGON* ou *MULTIPOINT* pour la 2D, ou *POINTZ*, *ARCZ*, *POLYGONZ* ou *MULTIPOINTZ* pour la 3D. Les shapefiles avec des valeurs de mesure ne sont pas gérés, ni les fichiers *MULTIPATCH*.
- **ENCODING=value :** définie la valeur de l'encodage dans le fichier DBF. La valeur par défaut est "LDID/87". Il n'est pas clair quelles autres valeurs peuvent être appropriées.

G Exemples

Un merge de deux shapefile *file1.shp* et *file2.shp* dans un nouveau fichier *file_merged.shp* est réalisée de cette manière :

```
% ogr2ogr file_merged.shp file1.shp
% ogr2ogr -update -append file_merged.shp file2.shp -nln file_merged
```

La seconde commande ouvre le fichier file_merged.shp en mode 'mise à jour' et tente de trouver des couches existantes et d'ajouter des géométries en copie.

L'option *-nln* définie le nom de la couche à copier.

Lisez également :

- Shapelib Page
- ESRI Shapefile Technical Description
- Notes utilisateurs sur le pilote Shapefile d'OGR <http://trac.osgeo.org/gdal/wiki/UserDocs/Shapefiles>`_

Chapitre CXXXIV SQLite/SpatiaLite RDBMS

OGR gère en option les tables spatiales et non-spatiale stocké dans les fichiers de base de données SQLite. SQLIte est un moteur RDBMS basé sur des fichiers simples et légers avec des commandes SQL assez complètes et avec des performances respectables.

Le pilote recherche une table *geometry_columns* énoncés tels que définis par le standard *Simple Features* de l'OGC, plus particulièrement comme définie dans la RFC 16 de FDO. Si trouvé, elle est utilisé pour mappé les tables et les couches.

Si *geometry_columns* est trouvée, chaque table est traitée comme une couche. Les couches avec un champ *WKT_GEOMETRY* seront traitées comme des tables spatiales, et la colonne *WKT_GEOMETRY* sera lu comme une géométrie Well Known Text

Si *geometry_columns* est trouvée, elle sera utilisée pour rechercher le système de référence spatiale dans la table *spatial_ref_sys*.

Les bases de données SQLite sont essentiellement sans type, mais le pilote SQLite tentera de classifier les champs attributaires comme texte, entier ou virgule flottante basé sur le contenu de la première ligne de la table. Aucun type de champ attributaire n'existant dans SQLite.

Les bases de données SQLite fonctionne souvent mal sur un NFS, ou d'autres protocoles de systèmes de fichiers par réseau dû à une faible gestion des cadenas. Il est plus sure d'opérer seulement avec des fichiers SQLite sur un disque système du système local.

SQLite est un pilote compilé en option. Il n'est pas compilé par défaut.

Bien que le pilote SQLite gère la lecture des données spatiales à partir des enregistrements, il n'y a pas de gestion pour l'indexation spatiale, donc les requêtes spatiales tendent à être longue. Les requêtes attributaires peuvent être rapide, spécialement si des indexes sont construits pour les colonnes attributaires appropriées en utilisant la commande SQL "CREATE INDEX ON ( )".

Par défaut, les requêtes SQL sont passées directement au moteur de base de données SQLite. Il est également possible de demander au pilote de prendre en charge les commandes SQL avec le :ref:`gdal.ogr.sql`, en passant la chaine "OGRSQL" à la méthode *ExecuteSQL()*, comme nom du dialecte SQL.

Unknown interpreted text role "ref".

À partir de OGR 1.8.0, l'option de configuration *OGR_SQLITE_SYNCHRONOUS* a été ajoutée. Quand définie à OFF, cela déclenche une commande 'PRAGMA synchronous = OFF' à la base de données SQLite. Cela a l'avantage d'accélérer certaines opérations d'écriture (par exemple les systèmes de fichier EXT4), mais au dépend de la sécurité des données ; crash du système/OS. Utilisez le donc avec soin dans des environnements en production et lisez la documentation de SQLite qui en parle.

A Utiliser la bibliothèque SpatiaLite (extension spatiale pour SQLite)

(À partir de GDAL 1.7.0)

Le pilote SQLite peut lire et écrire des bases de données SpatiaLite sans nécessité le chargement de la bibliothèque SpatiaLite. Mais si vous configurez GDAL/OGR avec un lien explicite à la bibliothèque SpatiaLite (version >= 2.3), vous aurez les avantages de toutes les extensions fournie par cette bibliothèque, comme les indexes spatiaux, les fonctions spatiales, etc.

Quelques exemples :

```
# Duplique la base de données échantillon fournie avec SpatiaLite (ne
nécessite
# pas un lien explicite avec SpatiaLite)
ogr2ogr -f SQLite testspatialite.sqlite test-2.3.sqlite  -dsco
SPATIALITE=YES

# Ajouter un index spatial sur la colonne géométrique de la table
Towns
# (nécessite un lien explicite avec SpatiaLite)
ogrinfo testspatialite.sqlite -sql "SELECT CreateSpatialIndex('Towns',
'geometry')"

# Faire une requête avec un filtre spatial (nécessite un lien
explicite avec SpatiaLite)
ogrinfo testspatialite.sqlite \
    -sql "SELECT Name, asText(geometry) FROM Towns WHERE
MBRIntersects(geometry, BuildMBR(754000, 4692000, 770000, 4924000))"
```

ou

```
# fonctionnera plus rapidement avec des filtres spatiaux et des liens
# explicites avec SpatiaLite
ogrinfo testspatialite.sqlite Towns -spat 754000 4692000 770000
4924000
```

B Problèmes de création

Le pilote SQLite gère la création de fichiers de base de données SQLite ou l'ajout de tables à une base existante. Remarquez qu'un nouveau fichier de base de données ne peut pas être crée sur un fichier existant.

1 Options de création de la base de données

- **METADATA=yes/no :** cela permet d'éviter la création des tables *geometry_columns* et *spatial_ref_sys* dans une nouvelle base de données. Par défaut les tables de méta-données sont créées lorsqu'une nouvelle base de données est crée.
- **SPATIALITE=yes/no :** (à partir de GDAL 1.7.0) créé le mode SpatiaLite des tables de métadonnées, qui sont un peu différente des métadonnées utilisées par ce pilote OGR et des spécifications OGC. Implique **METADATA=yes**.
- **INIT_WITH_EPSG=yes/no :** (à partir de GDAL 1.8.0) insère le contenu du fichier CSV EPSG dans la table *spatial_ref_sys*. no par défaut.

2 Options de création de couche

- **FORMAT=WKB/WKT/SPATIALITE :** contrôle le format utilisé pour la colonne géométrique. Par défaut WKB (Well Known Binary) est utilisé. Cela est généralement plus efficace lors du traitement et de l'espace utilisé, mais plus complexe à comprendre ou utiliser dans des application simples que le WKT (Well Known Text). L'extension SpatiaLite utilise son propre format binaire pour stocker les géométries et vous pouvez le choisir également. Il sera automatiquement sélectionné lorsque la base de données SpatiaLite est ouverte ou créé avec l'option **SPATIALITE=yes**. La valeur SPATIALITE est disponible à partir de GDAL 1.7.0
- **LAUNDER=yes/no :** contrôle si les noms de la couche et des champs seront nettoyer pour une utilisation plus facile dans SQLite. Les noms nettoyés seront converties en minuscule et certains caractères spéciaux seront changés en 'underscores'.
- **SPATIAL_INDEX=yes/no :** (À partir de GDAL 1.7.0) si la base de données est de type SpatiaLite, et si OGR est lié à libspatialite, cette option peut être utilisé pour contrôler si un index spatiale doit être créé. Yes par défaut.

C Autres informations

- Le développement du pilote SQLite pour OGR a été financé par le groupe DM Solutions et GoMOOS.
- Page principale pour SQLite.
- Extension spatiaLite de SQLite.
- FDO RFC 16 : Fournisseur FDO pour SQLite

Chapitre CXXXV SUA - Tim Newport-Peace's Special Use Airspace Format

(GDAL/OGR >= 1.8.0)

Ce pilote lit les fichiers décrivant le format .SUA des Special Use Airspaces dans le Tim Newport-Peace.

Airspace sont retournée comme features d'une couche unique, avec une géométrie de type Polygon et les champs suivants : TYPE, CLASS, TITLE, TOPS, BASE.

Les géométries Airspace faites d'arcs seront téssélées.

A Voir également

- Description du format .SUA

Chapitre CXXXVI SVG - Scalable Vector Graphics

(OGR >= 1.9.0)

OGR gère la lecture du format SVG (si GDAL a été compilé avec la gestion de la bibliothèque *expat*).

Pour le moment, il lit uniquement les fichiers SVG qui sont renvoyé par le Serveur Vecteur en Flux de Cloudmade.

Toutes les coordonnées sont relative au SRS Pseudo-mercator (EPSG:3857).

Le pilote renverra 3 couches :

- points
- lines
- polygons

A Voir également

- Page SVG du W3C
- Documentation du vecteur de Cloudmade

Chapitre CXXXVII U.S. Census TIGER/Line

Les ensembles de fichier TIGER/Line sont gérés en lecture.

Les fichiers TIGER/Line sont un base de données numérique d'objet géographique, tels que les routes, les rails, les rivières, les lacs, les frontières politiques, les limites des recensements statistique, etc. couvrant entièrement les États-Unis. La base de données contient des informations sur ces objets tel que leur location en latitude et longitude, le nom, le type de l'objet, la précision de l'adresse pour la plupart à la rue, la relation géographique aux autres objets, et aux autres informations liées. Ce sont des produits publiques créés à partir des bases de données d'information géographique TIGER (*Topologically Integrated Geographic Encoding and Referencing*) du Bureau du Recensement. TIGER a été développé au Bureau du Recensement pou gérer la cartographie et les activités liées à la géographie nécessaire par le recensement décennal et les programmes de sondage. Notez que les produits TIGER/Line n'inclut pas de statistique du recensement démographique. Ceux-ci sont vendu par le Bureau de Recensement dans un format séparé (pas directement géré par FME), mais ces statistiques sont liés aux blocs *census* dans les fichiers TIGER/Line.

Pour ouvrir un jeu de données TIGER/Line, sélectionnez le répertoire contenant un ou plusieurs ensembles de fichiers de données. Les étendues sont des régions, ou équivalent à une région. Chaque région consiste à une série de fichier avec un nom de base, et différentes extensions. Par exemple, la région 1 dans l'état 26 (Michigan) consiste à l'ensemble des fichiers suivants définie dans Tiger98.

```
TGR26001.RT1
TGR26001.RT2
TGR26001.RT3
TGR26001.RT4
TGR26001.RT5
TGR26001.RT6
TGR26001.RT7
TGR26001.RT8
TGR26001.RT9
TGR26001.RTA
TGR26001.RTC
TGR26001.RTH
TGR26001.RTI
```

```
TGR26001.RTP
TGR26001.RTR
TGR26001.RTS
TGR26001.RTZ
```

Le système de coordonnées de TIGER/Line est codé en dur à NAD83 lat/long, degré. Cela doit être approprié pour toutes les années récentes de la production TIGER/Line.

Il n'y a pas de gestion de mise à jour ou de création dans le pilote TIGER/Line.

Le lecteur a été implémenté pour les fichiers TIGER/Line de 1998, mais des développements ont permis de s'assurer de la compatibilité avec les produits TIGER/Line de 1992, 1995, 1997, 1999, 2000, 2002, 2003 et 2004. Les produits 2005 fonctionne également correctement selon des retours. Tous les produits TIGER/Line à partir de 1988 devrait fonctionner avec le lecteur, avec des pertes d'informations spécifique à certaine version.

A Représentation des objets

Avec quelques exceptions, un objet est créé pour chaque enregistrement d'un fichier de données TIGER/Line. Chaque fichier (c'est à dire, .RT1, .RTA) est traduit en un type d'objet OGR approprié, avec des noms d'attributs correspondant à ceux dans le manuel du produit TIGER/Line.

Les attributs *RT* (*record type*) et *VERSION* de TIGER/Line sont généralement ignorés, mais l'attribut *MODULE* est ajouté à chaque objet. L'attribut *MODULE* contient le nom de base (par exemple *TGR26001*) du module région à partir duquel est originaire l'objet. Pour certaines clés (telles que *LAND*, *POLYID*, et *CENID*)cet attribut *MODULE* est nécessaire pour rendre la clé unique quand le jeu de données (répertoire) consiste à des données de plus d'une région.

Ce qui suit est une liste de types d'objet, et leur lien avec le produit TIGER/Line.

1 CompleteChain

Une *CompleteChain* est une polyligne avec un TLID associé (TIGER/Line ID). Les objets *CompleteChain* sont établi à partir d'un enregistement de type 1 (Enregistrement de Données Basiques de Chaine Complète, *Complete Chain Basic Data Record*),, et is disponible est associé avec un enregistrement de type 3 (Codes d'Entité Géographique de Chaîne Complète, *Complete Chain Geographic Entity Codes*).Également, n'importe quels enregistrements de type 2 (Coordonnées de Forme de Chaîne Complète, *Complete Chain Shape Coordinates*) disponible sont utilisés pour remplir les points de forme intermédiaire sur l'arc. Le TLID est la clé primaire, et est unique dans la couverture nationale entière de TIGER/Line.

Ces objets ont toujours une géométrie 'line'.

2 AltName

Ces objets sont dérivés des enregistrement de type 4 (Index pour Alterner des Identifiants d'objet, *Index to Alternate Feature Identifiers*), et liés à un TLID de 1 à 4 numéros de noms d'objet (l'attribut *FEAT*) qui sont gardé séparément comme objets *FeatureIds*. La pipeline du lecteur standard attache le nom à partir des objets *FeatureIds* comme des attributs array *ALT_FEDIRS{}*, *ALT_FEDIRP{}*, *ALT_FENAME{}* et

ALT_FETYPE{}. *ALT_FENAME{}* est une liste de noms d'objets associée avec le TLID sur l'objet *AltName*.

Notez que zéro, un ou plusieurs enregistrement *AltName* peu(ven)t exister pour un TLID dpnnée, et chaque enregistrement *AltName* peut contenir entre un et quatre noms alternatifs. Parce que cela est encore très difficile d'utiliser les objets *AltName* pour lier des noms alternatifs à *CompleteChains*, il est anticipé que la pipeline du lecteur standard pour les fichiers TIGER/Line seront mis à jour dans un futur proche, entrainant la simplification des noms alternatifs.

Ces objets n'ont pas de géométrie associée.

3 FeatureIds

Ces objets sont dérivés des enregistrement du type 5 (Identifiants d'Objet de chaîne Complète, *Complete Chain Feature Identifiers*). Chaque objet contient un nom d'objet (*FENAME*), et son code d'identifiant d'objet associé (*FEAT*). L'attribut *FEAT* est la clé primaire, et est unique dans le module région. *FeatureIds* possède une relation un à plusieurs avec les objets *AltName* et *KeyFeatures*.

Ces objets n'ont pas de géométrie associée.

4 ZipCodes

Ces objets sont dérivés des enregistrements detype 6 (Données des Codes Postaux et des Précisions d'Adresse Aditionnelle, *Additional Address Range and ZIP Code Data*). Ces objets ont pour objectif d'augmenter les informations du Code ZIP gardé directement dans les objets *CompleteChain*, et il y a une relation plusieurs à un entre les objets *ZipCodes* et *CompleteChain*.

Ces objets n'ont pas de géométrie associée.

5 Landmarks

Ces objets sont dérivés des enregistrements de type 7 (Objets Landmark, *Landmark Feature*). Ils sont liés à un point ou à une zone de repère (*landmark*). Pour les zones de repère il y a une relation de un à un avec un enregistrement *AreaLandmark*. L'attribut *LAND* est une clé primaire et unique dans un module région.

Ces objets peuvent avoir une géométrie ponctuelle associée. Les points de repères associés avec des polygones n'auront pas la géométrie polygonale attachée. Il sera nécessaire de le collecter (via l'objet *AreaLandmark*) à partir d'un objet Polygone.

6 AreaLandmarks

Ces objets sont dérivé des enregistrement de type 8 (Polygone lié aux zone de repérage, *Polygons Linked to Area Landmarks*). Chacun associé un objet landmark (attribut *LAND*) avec un objet polygone (attribut *POLYID*). Cet objet a une relation plusieurs à plusieurs avec les objets polygones.

Ces objets n'ont pas de géométrie associée.

7 KeyFeatures

Ces objets sont dérivés des enregistrements de type 9 (Codes d'entité Géographique des

Polygones, *Polygon Geographic Entity Codes*). Ils peuvent être associé avec un objet *FeatureIds* (via l'attribut *FEAT*), et un objet polygone (via l'attribut *POLYID*).

Ces objets n'ont pas de géométrie associée.

8 Polygon

Ces objets sont dérivées des enregistrements de type A (Codes d'Entité des Polygones Géographiques, *Polygon Geographic Entity Codes*) et si disponible le type S relié (Codes d'Entité Additionnel Géographique, *Polygon Additional Geographic Entity Codes*). L'attribut *POLYID* est la clé primaire, identifiant d'une manière unique un polygone dans un module *country*.

Ces objets n'ont pas de géométrie associée avec eux puisqu'il est lu par le pilote TIGER d'OGR. Il doit être lié en externe en utilisant le *PolyChainLink*. Le script ''gdal/pymod/samples/tigerpoly.py'' peut être utilisé pour lire un jeu de données TIGER et extraire la couche polygone avec une géométrie comme un shapefile.

9 EntityNames

Ces objets sont dérivées des enregistrements de type C (Nomes d'Entité Géographique).

Ces objets n'ont pas de géométrie associée.

10 IDHistory

Ces objets sont dérivés des enregistrements de type H (Historique des ID de TIGER/LINE, *TIGER/Line ID History*). Ils peuvent être utilisés pour tracer le découpage, la fusion, la création et la suppression des objets *CompleteChain*.

Ces objets n'ont pas de géométrie associée.

11 PolyChainLink

Ces objets sont dérivés des enregistrement de type I (Liens entre les Polygones et les Chaines Complète, *Link Between Complete Chains and Polygons*). Ils sont normalement tous consommés par la pipeline du lecteur standard pendant le rattachement des géométries *CompleteChain* aux objets polygones pour établir leur géométries polygonales. Les objets *PolyChainLink* ont une relation plusieurs à un avec les objets polygones, et une relation d'un à un avec les objets *CompleteChain*.

Ces objets n'ont pas de géométrie associée.

12 PIP

Ces objets sont dérivés des enregistrements de type P (Point interne à un Polygone, *Polygon Internal Point*). Ils sont reliés à un objet Polygone vie l'attribut *POLYID*, et peuvent être utilisé pour établir un point interne pour les objets polygones.

Ces objets n'ont pas de géométrie associée.

13 ZipPlus4

Ces objets sont dérivés des enregistrements de type Z (Codes ZIP + 4). Les objets *ZipPlus4* ont une relation plusieurs à un avec les objets *CompleteChain*.

Ces objets n'ont pas de géométrie associée.

B Voir également

- http://www.census.gov/geo/www/tiger/ : Plus d'information sur le format de fichier TIGER/Line, et les produits de données peuvent être trouvés sur cette page web de Census US.

Chapitre CXXXVIII VFK - format de données d'échange cadastrale Tchèque

Le pilote VFK peut lire les données dans le *format de données d'échange cadastrale Tchèque*. Le fichier VFK est reconnu comme une source de données OGR avec zéro ou plusieurs couches OGR.

Les points sont représenté en tant que wkbPoints, lines et boundaries en tant que wkbLineStrings et les surfaces en tant que wkbPolygons. Les primitives wkbMulti* ne sont pas utilisées. Les types de features ne peuvent pas être mixés dans une couche.

A Nom de la source de données

Le nom de la source de données est un chemin complet vers le fichier VFK.

B Noms des couches

Les blocs de données VFK sont utilisés comme noms de couches.

C Filtre attributaire

Le moteur SQL interne d'OGR est utilisé pour évaluer l'expression. L'évaluation est faites une fois lorsque le filtre attributaire est définie.

D Filtre spatial

Bounding boxes des features stockées dans la structure topologique sont utilisés pour évaluer si une feature correspond au filtre spatial en cours. L'évaluation est faites une fois que le filtre spatiale est définie.

E Références

- Problèmes d'implémentation du pilote VFK d'OGR
- Documentation du format de données d'échange Tchèque (en Tchèque)

Chapitre CXXXIX Virtual Format

Le format Virtuel d'OGR est un pilote qui transforme les objets lu à partir d'autres pilotes sur des critères définie dans un fichier de contrôle XML. Il est d'abord utilisé pour profiter des couches spatiales provenant de tables à plat avec des informations spatiales dans des colonnes attributaires. Il peut être également utilisé pour associer des information de système de coordonnées avec une source de données, merger des couches de sources de données différentes dans une seule source de données, ou même juste fournir un fichier de soutient pour l'accès à une source de données pas sous forme de fichier.

Les fichiers virtuel sont pour l'instant normalement préparés à la main.

A Problèmes de création

Avant GDAL 1.7.0, le pilote VRT d'OGR était en lecture seule.

Depuis GDAL 1.7.0, les opération *CreateFeature()*, *SetFeature()* et *DeleteFeature()* sont gérées sur la couche d'un jeu de données VRT, si les conditions suivantes sont remplies :

- le jeu de données VRT est ouvert en mode update ;
- la couche source sous-jascente gère ces opérations ;
- l'élément *SrcLayer* est utilisé (en opposition à l'élément *SrcSQL*) ;
- le FID des features VRT est le même que le FID des features sources, c'est à dire, l'élément *FID* n'est pas définie.

B Format de fichier virtuel

L'élément racine d'un fichier de contrôle XML est *OGRVRTDataSource*. il a un enfant *OGRVRTLayer* pour chaque couche dans la source de données virtuelle. Cet élément doit avoir un attribut **name** avec le nom de la couche, et peut avoir les sous-éléments suivants :

- **SrcDataSource (obligatoire) :** La valeur est le nom de la source de données dont cette couche sera dérivée. L'élément peut en option avoir un attribut *relativeToVRT* dont la valeur par défaut est "0", mais "1" indique que la source de données doit être interprétée comme relative au fichier virtuel. Cela peut être un jeu de données géré par OGR, incluant ODBC, CSV, etc. L'élément peut également avoir un attribut partagé pour contrôler si la source de données doit

être ouverte en mode partagé. OFF par défaut pour l'utilisation de *SrcLayer* et ON pour l'utilisation de *SrcSQL*.

- **SrcLayer (optionnel) :** la valeur est le nom de la couche dans la source de données dont cette couche virtuelle sera dérivée. Si cet élément n'est pas fournit, alors l'élément *SrcSQL* doit être fournit.
- **SrcSQL* (optionnel) :** une requête SQL pour exécuter la génération de la couche désirée résultante. Ceci doit être fournie à la place de *StcLayer* pour les résultats dérivés de requêtes. certaines limitations peuvent s'appliquer pour les couches dérivées du SQL.
- **FID (optionnel) :** nom de la colonne attributaire à partir duquel le FID . des objets doivent être dérivés. S'il n'est pas fourni, le FID des objets sources sera utilisé directement.
- **Style (optionnelle) :** nom de la colonne attributaire à partir duquel le style des features doit être dérivé. Si non fournie, le style de la feature source sera utilisé directement.
- **GeometryType (optionnel) :** le type de la géométrie à assigner à la couche. S'il n'est pas fournie il sera récupéré à partir de la couche source. La valeur doit être un parmi *wkbNone*, *wkbUnknown*, *wkbPoint*, *wkbLineString*, *wkbPolygon*, *wkbMultiPoint*, *wkbMultiLineString*, *wkbMultiPolygon, ou* **wkbGeometryCollection*". En option *25D* peut ajouté pour marquer l'utilisation des coordonnées Z. Par défaut *wkbUnknown* indique que n'importe quelle géométrie est autorisée.
- **LayerSRS (optionnel) :** la valeur de cet élément est la référence spatiale à utiliser pour la couche. Si elle n'est pas fournie, elle est héritée de la couche source. La valeur peut être en WKT ou dans n'importe quel autre format accepté par la méthode *OGRSpatialReference::SetUserInput()*. Si la valeur est NULL, alors aucun SRS ne sera utilisé pour la couche.
- **GeometryField (optionnel) :** cet élément est utilisé pour définir comment la géométrie pour les objets doit être dérivée. Si elle n'est pas fournie la géométrie de l'objet source est copié directement. Le type de l'encodage de la géométrie est indiqué avec l'attribut d'encodage qui peut avoir les valeurs *WKT*, *WKB* ou *PointFromColumns*. Si l'encodage est *WKT* ou *WKB* alors le champ attributaire aura le nom du champ contenu la géométrie WKT ou le WKB. Si l'encodage est *PointFromColumns* alors les attributs x, y et z aura les noms des colonnes à utilisées pour les coordonnées X, Y et Z. L'attribut z est optionnel. À partir de GDAL 1.7.0, l'attribut optionnel **reportSrcColumn** peut être utilisé pour définir si les champs géométriques sources (l'ensemble des champs dans les attributs **field**, **x**, **y** ou **z**) doivent être reporté comme champs de la couche vrt. TRUE par défaut. Si définie à FALSE, les champs géométriques sources seront utilisé seulement pour construire la géométrie de la feature de la couche VRT.
- **SrcRegion (optionnel, à partir de GDAL 1.7.0) :** Cet élément est utilisé pour définir un filtre spatial initial pour les features sources. Ce filtre spatial sera combiné avec n'importe quel filtre spatial explicitement définie sur la couche VRT avec la méthode *SetSpatialFilter()*. La valeur de l'élément doit être une chaîne WKT valide définissant un polygone. Un attribut *clip* optionnel peut être définie à "TRUE" pour découper les géométries à la région source, sinon les géométries sources ne sont pas modifiées.
- **Field (optionnel, à partir de GDAL 1.7.0) :** un ou plusieurs champs attributaires peuvent être définie avec les éléments Field. Si aucun élément Field ne sont définie, les champs de ma couche/sql source sera définie sur la couche vrt. L'élément Field peut avoir les attributs suivants :
  - **name (nécessaire) :** le nom du champs.

- **type :** le type du champs, un parmi "Integer", "IntegerList", "Real", "RealList", "String", "StringList", "Binary", "Date", "Time", ou "DateTime" - "String" par défaut.
- **width :** la largeur du champ, inconnus par défaut.
- **precision :** la précision du champs, 0 par défaut.
- **src :** le nom du champ source qui doit être copié dans celui-ci. par défaut, la valeur de "name".

C Exemple : couche ponctuelle ODBC

Dans l'exemple suivant (disease.ovf) la mauvaise table à partir de la base de données ODBC DISEASE est utilisée pour créer une couche spatiale. Le fichier virtuel utilise les colonnes "x" et "y" pour obtenir la localisation spatiale. La couche est également définie comme une couche point, et comme étant dans le système de coordonnées WGS84.

```
<OGRVRTDataSource>

    <OGRVRTLayer name="worms">
        <SrcDataSource>ODBC:DISEASE,worms</SrcDataSource>
        <SrcLayer>worms</SrcLayer>
        <GeometryType>wkbPoint</GeometryType>
        <LayerSRS>WGS84</LayerSRS>
        <GeometryField encoding="PointFromColumns" x="x" y="y"/>
    </OGRVRTLayer>

</OGRVRTDataSource>
```

D Exemple : renommer des attributs

Il peut être utile dans certaines circonstance de pouvoir renommer les noms des champs à partir d'une couche source en un nom différent. Cela est particulièrement vrai quand on veut traduire vers un format dont les schéma est imposé, tel que le format GPX (<name>, <desc>, etc.). Cela peut être accomplit en utilisant SQL de cette manière :

```
<OGRVRTDataSource>
    <OGRVRTLayer name="remapped_layer">
        <SrcDataSource>your_source.shp</SrcDataSource>
        <SrcSQL>SELECT src_field_1 AS name, src_field_2 AS desc FROM
your_source_layer_name</SrcSQL>
    </OGRVRTLayer>
</OGRVRTDataSource>
```

Cela peut aussi être accomplie (à partir de GDAL 1.7.0) en utilisant des définitions de champs explicites :

```
<OGRVRTDataSource>
    <OGRVRTLayer name="remapped_layer">
        <SrcDataSource>your_source.shp</SrcDataSource>
        <SrcLayer>your_source</SrcSQL>
        <Field name="name" src="src_field_1" />
        <Field name="desc" src="src_field_2" type="String"
width="45" />
    </OGRVRTLayer>
```

```
</OGRVRTDataSource>
```

E Exemple : Filtre spatial transparent (GDAL >= 1.7.0)

L'exemple suivant retournera seulement les features à partir de la couche source qui intersecte la région (0,40)-(10,50). De plus, les géométries retournées seront découpées pour correspondre à cette région.

```
<OGRVRTDataSource>
    <OGRVRTLayer name="source">
        <SrcDataSource>source.shp</SrcDataSource>
        <SrcRegion clip="true">POLYGON((0 40,10 40,10 50,0 50,0
40))</SrcRegion>
    </OGRVRTLayer>
</OGRVRTDataSource>
```

F Autres remarques

- Quand *GeometryField* est *WKT*, les filtres spatiaux sont appliqués après extractions de toutes les lignes à partir de la source de données. Essentiellement, cela signifie qu'il n'y a pas de filtre spatial rapide sur les géométries dérivées WKT.
- Quand *GeometryField* est *PointFromColumns*, et qu'un *SrcLayer* (en opposition à *SrcSQL*) est utilisé, et qu'un filtre spatial est effectif sur la couche virtuelle alors le filtre spatial sera traduit en interne en un filtre attribut sur les colonnes X et Y dans *SrcLayer*. Au cas où des filtres spatiaux rapide soient importants, il peut être utile d'indexer les colonnes X et Y dans le stockage des données source, si cela est possible. Par exemple si la source est un RDBMS. Vous pouvez désactiver cette fonctionnalité en définissant l'attribut *useSpatialSubquery* de l'élément GeometryField à FALSE.
- Normalement la *SrcDataSource* est au format tabulaire non-spatial (tel que MySQL, SQLite, CSV, OCI, ou ODBC) mais il peut être également une base de données spatiales auquel cas la géométrie peut être directement copiée.

Chapitre CXL WFS - Service WFS OGC

(GDAL/OGR >= 1.8.0)

Ce pilote peut se connecter à un service WFS de l'OGC. Il gère les protocoles WFS 1.0.0 et WFS 1.1.0. GDAL/OGR doit être compilé avec la gestion de Curl dans le but de compiler le pilote WFS. Habituellement les requêtes WFS renvoie les résultats au format GML, le pilote GML doit donc généralement être définie pour la gestion de la lecture (nécessite donc que GDAL/OGR soit compilé avec la gestion de Xerces ou Expat). Il est parfois possible d'utiliser un formats sous-jascent quand le serveur les gère (tel que OUTPUTFORMAT=json).

Le pilote gère les services en lecture seul, ainsi que ceux en mode Transactionnel (WFS-T).

A Syntaxe des noms de jeux de données

La syntaxe minimale pour ouvrir une source de données WFS est : *WFS:http://path/to/WFS/service* ou *http://path/to/WFS/service?SERVICE=WFS*

Des paramètres optionnels additionnels peuvent être définie tels que *TYPENAME*, *VERSION*, *MAXFEATURES* comme spécifié dans la spécification WFS.

Il est également possible de définir le nom d'un fichier XML dont le contenu correspond à la syntaxe suivante (l'élément <OGRWFSDataSource> doit être le premier octet du fichier) :

```
<OGRWFSDataSource>
    <URL>http://path/to/WFS/service[?
OPTIONAL_PARAMETER1=VALUE[&amp;OPTIONNAL_PARAMETER2=VALUE]]</URL>
</OGRWFSDataSource>
```

À la première ouverture, le contenu du résultat de la requête *GetCapabilities* sera ajouté au fichier, afin de mettre en cache pour une ouverture ultérieure du jeu de données. La même chose s'applique pour la requête *DescribeFeatureType* réalisée pour découvrir la définition des champs de chaque couche.

Le fichier de description de service possède les éléments additionnels suivants comme enfant immédiat de l'élément `OGRWFSDataSource` qui peuvent être définie en option.

- **Timeout :** la valeur du timeout à utiliser pour les requêtes des services distants. Si non définie la valeur par défaut de libcurl est utilisée.
- **UserPwd :** peut fournir un couple *userid:password* pour envoyer un userid et un mot de passe au serveur distant.
- **HttpAuth :** peut être BASIC, NTLM ou ANY pour contrôler la méthode d'authentification à utiliser.
- **Version :** définie une version spécifique du WFS à utiliser (soit 1.0.0 ou 1.1.0).
- **PagingAllowed :** définir à ON pour que la pagination soit activée. Voir la section :ref:`gdal.ogr.formats.wfs.pagination`.
  Unknown interpreted text role "ref".
- **PageSize :** taille de la page quand la pagination est activée. Voir la section :ref:`gdal.ogr.formats.wfs.pagination`.
  Unknown interpreted text role "ref".

B Pagination des requêtes

Généralement, lors de la lecture de la première feature d'une couche, le contenu de la couche entière sera récupérée du serveur.

Certains serveurs (comme MapServer >= 6.0) gère une option spécifique du fournisseur, STARTINDEX, qui permet de faire une requête par "page", et donc d'éviter le téléchargement de tout le contenu de la couche en une seule requête. Le client WFS d'OGR utilisera la pagination quand l'option de configuration *OGR_WFS_PAGING_ALLOWED* est définie à ON. La taille de la page (nombre de feature récupérée en une seule requête) est limité à 100 par défaut. Elle peut être changée en définissant l'option de configuration *OGR_WFS_PAGE_SIZE*. Ces deux options peuvent également être définie dans le fichier XML de description WFS.

C Filtrage

Le pilote renverra n'importe quel filtre spatial avec *SetSpatialFilter()* vers le serveur. Il fera également sont possible pour les filtres attributaires définie avec *SetAttributeFilter()* (traduire le langage SQL d'OGR en description du filtre OGC). Quand cela n'est pas possible, il sera par défaut un filtre côté client, ce qui peut être une opération lente parce qu'impliquant la récupération de toutes les features du serveur.

D Gestion de l'écriture / WFS-T

Le protocol WFS-T permette à l'utilisateur d'opérer au niveau de la feature. Aucune création de source de données, couche ou champs n'est pas possible.

La gestion de l'écriture est seulement activé lorsque la source de données est ouverte en mode update.

Les correspondances entre les opération du service e transaction WFS et les concepts OGR sont les suivantes :

- OGRFeature::CreateFeature() <==> opération d'insertion WFS
- OGRFeature::SetFeature() <==> opération de mise à jour WFS
- OGRFeature::DeleteFeature() <==> opération de suppression WFS

Les opérations de locks (service LockFeature) ne sont pas disponible pour le moment.

Il a quelques contraintes à garder à l'esprit. L'ID des features (FID) d'OGR est un entier, tandis que l'attribut gml:id du WFS/GML est une chaîne de caractères. Il n'est donc pas

toujours possible de correspondre les deux valeurs. Le pilote WFS expose alors l'attribut gml:id d'une feature comme un champ 'gml_id'.

Lors de l'insertion d'un nouvelle feature avec *CreateFeature()*, et si la commande est réussie, OGR récupérera le gml:id et définira le champ 'gml_id' de la feature en conséquence. Il tentera également de définir le FID OGR si le gml:id est de la forme *layer_name.numeric_value*. Sinon le FID sera laissé à sa valeur par défaut.

Lors de la mise à jour d'une feature existante avec *SetFeature()*, le champ FID OGR sera ignoré. La requête renvoyée au pilote prendra seulement en compte la valeur du champ gml:id de la feature. La même chose s'applique pour *DeleteFeature()*.

E Transaction OGR et gestion de l'écriture

Les opérations ci-dessus sont par défaut déclenchées vers le serveur en synchrone avec l'appel de l'API d'OGR. Cela peut cependant causer des problèmes de performances lorsque plusieurs commandes sont déclenchées dû à un grand nombre d'échanges client/serveur.

Il est possible de contourner ces opérations entre *OGRLayer::StartTransaction()* et *OGRLayer::CommitTransaction()*. Les opérations seront stockées en mémoire et seulement exécuté au moment où *CommitTransaction()* est appelé.

La contrainte de *CreateFeature()* est que l'utilisateur ne peut pas connaître quel gml:id a été assigné pour la feature insérée. Une requête spatiale SQL a été introduite dans le pilote WFS pour contourner ceci : en déclenchant la commande "`SELECT _LAST_INSERTED_FIDS_ FROM layer_name`" (où *layer_name* doit être remplacé par le *layer_name* réel) via *OGRDataSource::ExecuteSQL()*, une couche sera renvoyée avec autant de ligne avec un attribut unique gml_id que de features insérées pendant la dernière transaction commitée.

Note!

pour le moment, seulement CreateFeature() utilise le mécanisme de transaction d'OGR. SetFeature() et DeleteFeature() seront toujours déclenché immédiatement.

F Commandes SQL spéciales

Les commandes SQL et pseudo-SQL suivantes envoyées à *OGRDataSource::ExecuteSQL()* sont spécifiques au pilote WFS :

- "DELETE FROM layer_name WHERE expression" : cela résultera en une opération WFS de suppression. Cela peut être un moyen rapide de suppression d'une ou plusieurs features. En particulier, cela peut être un remplaçant plus rapide pour *OGRLayer::DeleteFeature()* quand gml:id est connu, mais la feature n'est pas récupéré à partir du serveur.
- "SELECT _LAST_INSERTED_FIDS_ FROM layer_name" : voir le paragraphe au-dessus.

Pour le moment, toutes les autres commandes SQL sera réalisée par la couche générique, c'est à dire seulement réalisée côté client. Les filtres spatiaux et attributaires côté serveur doit être réalisé via les interfaces *SetSpatialFilter()* et *SetAttributeFilter()*.

G Métadonnées des couches

(OGR >= 1.9.0)

Une couche cachée appelée "WFSLayerMetadata" est rempli avec les enregistrements des métadonnées pour chaque couche WFS.

Chaque enregistrement contient un champ "layer_name", "title" et "abstract", à partir du document renvoyé par le GetCapabilities.

Cette couche est retournée via GetLayerByName("WFSLayerMetadata").

H Exemples

- Liste les types d'un serveur WFS :

```
ogrinfo -ro WFS:http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
```

- Liste les types d'un serveur WFS dont la structure des couches sont en cache dans un fichier XML :

```
ogrinfo -ro mswfs_gmap.xml
```

- Liste les features d'une couche popplace, avec un filtre spatial :

```
ogrinfo -ro WFS:http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
popplace -spat 0 0 2961766.250000 3798856.750000
```

- Récupère les features de gml:id "world.2" et "world.3" à partir de la couche tows:world :

```
ogrinfo "WFS:http://www.tinyows.org/cgi-bin/tinyows" tows:world
-ro -al -where "gml_id='world.2' or gml_id='world.3'"
```

- Affiche la couche metadata (OGR >= 1.9.0):

```
ogrinfo -ro -al "WFS:http://v2.suite.opengeo.org/geoserver/ows"
WFSLayerMetadata
```

I Voir également

- Standard WFS de l'OGC
- :ref:`gdal.ogr.formats.gml`
  Unknown interpreted text role "ref".

Chapitre CXLI X-Plane/Flightgear aeronautical data

(disponible depuis GDAL 1.6.0)

Les données aéronautique X-Plane sont gérées en lecture seule. Ces données sont utilisées par exemple par le logiciel Flihgear et X-Plane.

Le pilote est capable de lire les fichiers suivants :

Nom du fichier	***Description***	***Versions gérées***
apt.dat	Données Aéroport	850, 810
nav.dat (ou earth_nav.dat)	Aides à la navigation	810, 740
fix.dat (ou earth_fix.dat)	Intersections IFR	600
awy.dat (ou earth_awy.dat)	Route de vol	640

Chaque fichier sera retourné comme un ensemble de couches dont le schéma de données est données ci-dessous. Le schéma des données est généralement aussi proche que possible que les schémas des données originelles décrit dans la spécification X-Plane. Cependant, vous noterez s'il vous plaît que les mètres (ou kilomètre) sont toujours utilisés pour retourner les hauteurs, les élévations, les distances (largeurs et longueurs), etc., même si les données originales sont parfois exprimées en pied ou en miles nautiques.

Les données sont retournées comme étant exprimé en datum WGS84 (latitude, longitude), bien que la spécification n'est pas très claire sur ce sujet).

L'option de configuration OGR_XPLANE_READ_WHOLE_FILE peut être définie à FALSE lors de la lecture d'un gros fichier en fonction de la RAM disponible (vrai tout particulièrement pour apt.dat). Cette option force le pilote à ne pas mettre en cache les features en RAM, mais juste à les récupérer de la couche en cours. Bien sur, cela aura un impact négatif sur les performances.

A Exemples

convertir toutes les couches contenu dans *apt.dat* dans un ensemble de shapefiles :

```
% ogr2ogr apt_shapes apt.dat
```

convertir toutes les couches contenu dans *apt.dat* dans une base de données PostgreSQL :

```
% PG_USE_COPY=yes ogr2ogr -overwrite -f PostgreSQL
PG:"dbname=apt" apt.dat
```

B Voir également

- Définition du fichier données X-Plane

C Airport data (apt.dat)

Ce fichier contient la description des éléments définissant les aéroports, les héliports, les bases nautiques, avec leur pistes et route de taxi, les fréquences ATC, etc.

Les couches suivantes sont retournées :

- APT (Point)
- RunwayThreshold (Point)
- RunwayPolygon (Polygone)
- WaterRunwayThreshold (Point)
- WaterRunwayPolygon (Polygone)
- Stopway (Polygon)
- Helipad (Point)
- HelipadPolygon (Polygone)
- TaxiwayRectangle (Polygone)
- Pavement (Polygone)
- APTBoundary (Polygone)
- APTLinearFeature (Line String)
- StartupLocation (Point)
- APTLightBeacon (Point)
- APTWindsock (Point)
- TaxiwaySign (Point)
- VASI_PAPI_WIGWAG (Point)
- ATCFreq (Aucun)

Toutes les couches autre que APT se réfèreront à l'aéroport grâce à la colonne *apt_icao*, qui peut servir de clé étrangère.

1 Couche APT

Principale description pour un aéroport. La position rapportée sera la position de la tour de contrôle si présente, autrement la position trouvée du premier seuil de la piste.

Champs :

- apt_icao : Chaine (4.0). Code *ICAO* de l'aéroport.
- apt_name : Chaine (0.0). Nom complet de l'aéroport.
- type: Integer (1.0). Airport type : 0 pour les aéroport régulier, 1 pour les bases d'hydravion, 2 pour les héliports (ajouté dans GDAL 1.7.0)
- elevation_m : Réel (8.2). Altitude de l'aéroport (en mètres).

- has_tower : Integer (1.0). Définie à 1 si l'aéroport a une tour de contrôle.
- hgt_tower_m : Réel (8.2). Hauteur de la tour de contrôle si présente.
- tower_name : Chaine (32.0). Nom de la tour de contrôle si présente.

1.a Couche RunwayThreshold

Cette couche contient la description d'un seuil d'une piste. La piste en elle-même est complètement décrite par ces deux seuils, et la couche *RunwayPolygon*.

Note!

quand une piste a un seuil déplacé, le seuil sera rapporté comme deux géométries : une à la position du seuil non déplacé (is_displaced=0), et l'autre à la position du seuil déplacé (is_displaced=1).

Champs :

- apt_icao : Chaine (4.0). code *ICAO* pour l'aéroport de ce seuil de piste.
- rwy_num : Chaine (3.0). Code pourla piste, tel que18, 02L, etc. Unique pour chaque aéroport.
- width_m : Réel (3.0). Largeur en mètre.
- surface : Chaine (0.0). Type de la surface parmi :
    - Asphalt
    - Concrete
    - Turf/grass
    - Dirt
    - Gravel
    - Dry lakebed
    - Water
    - Snow
    - Transparent
- shoulder : Chaine (0.0). Type d'accotement de la piste parmi :
    - None
    - Asphalt
    - Concrete
- smoothness : Réel (4.2). Douceur de la piste. Pourcentage entre 0.00 et 1.00. 1.25 est la valeur par défaut.
- centerline_lights : Entier (1.0). Définie à 1 si la piste a des lumières centrales
- edge_lighting : Chaine (0.0). Type de bord lumineux parmi :
    - None
    - Yes (quand importé des enregistrements V810)
    - LIRL : Lumière de piste de faible intensité(proposé pour V90x)
    - MIRL : Lumière de piste de moyenne intensité
    - HIRL : Lumière de piste de haute intensité (proposé pour V90x)
- distance_remaining_signs : Entier (1.0). Définie à 1 si la piste a des lumières de 'distance restante'.
- displaced_threshold_m : Réel (3.0). Distance entre le seuil et le seuil déplacé.
- is_displaced : Entier (1.0). Définie à 1 si la position est la position du seuil déplacé.
- stopway_length_m : Réel (3.0). Longueur de la piste d'arrêt/de la zone de décollage/de la zone de dépassement à la fin de l'approche de la piste en mètre.
- markings : Chaine (0.0). Marquage de la piste pour la fin de piste parmi :
    - None
    - Visual

        - Non-precision approach
        - Precision approach
        - UK-style non-precision
        - UK-style precision
- approach_lighting : Chaine (0.0). Lumière d'approche pour la fin de la piste parmi :
    - None
    - ALSF-I
    - ALSF-II
    - Calvert
    - Calvert ISL Cat II and III
    - SSALR
    - SSALS (Enregistrements V810)
    - SSALF
    - SALS
    - MALSR
    - MALSF
    - MALS
    - ODALS
    - RAIL
- touchdown_lights: Integer (1.0). Set to 1 if the runway has touchdown-zone lights (TDZL)
- REIL : Chaine (0.0). Lumière d'Identification de fin de piste, *Runway End Identifier Lights* (REIL) parmi :
    - None
    - Omni-directional
    - Unidirectionnal
- length_m : Réel (5.0). (Champ calculé). Longueur en mètre entre les deux seuils aux deux extrémités de la piste. Les seuils déplacés ne sont pas pris en compte dans ce calcul.
- true_heading_deg : Réel (6.2). (Champ calculé). En-tête réel en degré à l'approche de la fin de la piste.

1.b Couche RunwayPolygon

Cette couche contient la forme rectangulaire de la piste. Elle est calculé à partir des informations de seuil de la piste. Quand cela n'est pas définie, la signification du champ est la même que la couche *RunwayThreshold*.

Champs :

- apt_icao : Chaine (4.0)
- rwy_num1 : Chaine (3.0). Code pour le premier seuil de la piste. Par exemple *20L*.
- rwy_num2 : Chaine (3.0). Code pour le second seuil de la piste. Par exemple *02R*.
- width_m : Réel (3.0)
- surface : Chaine (0.0)
- shoulder : Chaine (0.0)
- smoothness : Réel (4.2)
- centerline_lights : Entier (1.0)
- edge_lighting : Chaine (0.0)
- distance_remaining_signs : Entier (1.0)
- length_m : Réel (5.0)
- true_heading_deg : Réel (6.2). En-tête réel de la première à la seconde piste.

1.c WaterRunwayThreshold (Point)

Champs :

- apt_icao : Chaine (4.0)
- rwy_num : Chaine (3.0). Code pour la piste, parexemple 18. Unique pour chaque aéroport.
- width_m : Réel (3.0)
- has_buoys : Entier (1.0). Définie à 1 si la piste doit être marqué avec des bouées flottantes sur l'eau.
- length_m : Réel (5.0). (Champ calculé) Longueur entre les deux extrémités de la piste amerrissage.
- true_heading_deg : Réel (6.2). (Champ calculé). En-tête réel en degré à l'approche de la fin de la piste.

1.d WaterRunwayPolygon (Polygone)

Cette couche contient la forme rectangulaire d'une piste d'ammerissage. Elle est construite à partir des informations des seuils des pistes d'atterrissage de l'eau.

Champs :

- apt_icao : Chaine (4.0)
- rwy_num1 : Chaine (3.0)
- rwy_num2 : Chaine (3.0)
- width_m : Réel (3.0)
- has_buoys: Integer (1.0)
- length_m : Réel (5.0)
- true_heading_deg : Réel (6.2)

1.e Stopway layer (Polygon)

(À partir de GDAL 1.7.0)

Cette couche contient la forme rectangulaire du prolongement d'arrêt qui peut être trouvé au début de la piste. C'est une partie du tarmac mais qui n'est pas censé être utilisée pour les opérations normales.

Il est calculé à partir des informations de longueur du prolongement d'arrêt de la piste et seulement présent si la longueur est différente de zéro.

Lorsqu'il n'est pas spécifié, la signification des champs est le même que pour la couche RunwayThreshold.

Champs :

- apt_icao: String (4.0)
- rwy_num: String (3.0).
- width_m: Real (3.0)
- length_m: Real (5.0) : longueur du prolongement de l'arrêt / du blastpad / du dépassement à la fin de l'approche de la piste en mètre.

1.f Helipad (Point)

Cette couche contient le centre de la piste d'atterrissage d'hélicoptères.

Champs :

- apt_icao : Chaine (4.0)

- helipad_name : Chaine (5.0). Nom de la piste d'atterrissage d'hélicoptères de la forme "Hxx". Unique pour chaque aéroport.
- true_heading_deg : Réel (6.2)
- length_m : Réel (5.0)
- width_m : Réel (3.0)
- surface : Chaine (0.0). Voyez ci-dessus pour les codes de surfaces des pistes.
- markings : Chaine (0.0). Voyez ci-dessus pour les codes de marquage des pistes.
- shoulder : Chaine (0.0). Voyez ci-dessus pour les codes d'accotement des pistes.
- smoothness : Réel (4.2). Voyez ci-dessus pour la description simple des pistes.
- edge_lighting : Chaine (0.0). Bord de piste d'atterrissage d'hélicoptère lumineux parmi :
    - None
    - Yes (Enregistrement V810)
    - Yellow
    - White (proposé pour V90x)
    - Red (Enregistrement V810)

1.g HelipadPolygon (Polygone)

Cette couche contient la forme rectangulaire d'une aire d'atterrissage d'hélicoptères. Les champs sont identique à la couche *Helipad*.

1.h TaxiwayRectangle (Polygone) - Enregistrement V810

Cette couche content la forme rectangulaire d'une voie de taxie.

Champs :

- apt_icao : Chaine (4.0)
- true_heading_deg : Réel (6.2)
- length_m : Réel (5.0)
- width_m : Réel (3.0)
- surface : Chaine (0.0). Voyez ci-dessus les codes des surfaces des pistes d'atterrissage.
- smoothness : Réel (4.2). Voyez ci-dessus la description douce des pistes d'atterrissage.
- edge_lighting : Entier (1.0). Définie à 1 si la piste de taxi a des bords lumineux.

1.i Pavement (Polygone)

Cette couche contient des tronçons polygonaux de chaussée pour les voies de taxi et des tabliers. Les polygones peuvent inclure des troues.

Le fichier source peut contenir des courbes de Béziers comme côté de polygone. Dû à un manque de gestion de telle géométrie dans le modele d'Objet Simple (Simple Feature) d'OGR, les coubres de Bézier sont discrétisées en morceaux linéaires.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)
- surface : Chaine (0.0). Voyez ci-dessus les codes des surfaces des pistes d'aviation.
- smoothness : Réel (4.2). Voyez ci-dessus la descriptions en douceur des pistes d'aviations.
- texture_heading : Réel (6.2). Direction du grain de texture de la chaussée en

degré réel

1.j APTBoundary (Polygone)

Cette couche contient les limites de l'aéroport. Il y a au maximum une telle géométrie par aéroport. Le polygone peut inclure des troues. Les courbes de béziers sont discrétisées en morceaux linéaires.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)

1.k APTLinearFeature (Chaine linéaire)

Cette couche contienr les objets linéaires. Les courbes de béziers sont d iscrétisées en morceaux linéaire.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)

1.l StartupLocation (Point)

Définie les positions des portes, locations de rampe, etc.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)
- true_heading_deg : Réel (6.2)

1.m APTLightBeacon (Point)

Définie les balises-lumières des aéroports.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)
- color : Chaine (0.0). Couleur de la lumière de la balise parmi :
    - None
    - White-green : aérogare
    - White-yellow : base d'hydravion
    - Green-yellow-white : héliports
    - White-white-green : champ militaire

1.n APTWindsock (Point)

Définie les biroutes des aéroports.

Champs :

- apt_icao : Chaine (4.0)
- name : Chaine (0.0)
- is_illuminated: Integer (1.0)

1.o TaxiwaySign (Point)

Définie les signes des voies de taxi des aéroports.

Champs :

- apt_icao : Chaine (4.0)
- text : Chaine (0.0). C'est d'une manière ou d'une autre encodé dans un format spécifique. Voyez les spécification d'X-Plane pour plus de détails.
- true_heading_deg : Réel (6.2)
- size: Integer (1.0). De 1 à 5. Voyez les spécification d'X-Plane pour plus de détails.

1.p VASI_PAPI_WIGWAG (Point)

Définie un *VASI*, *PAPI* ou *Wig-Wag*. Pour les valeurs *PAPI* et *Wig-Wags*, les coordonnées est le centre de l'affichage. Pour la valeur *VASI*, c'est le point central entre les deux unités de lumière des *VASI*.

Champs :

- apt_icao : Chaine (4.0)
- rwy_num : Chaine (3.0). Clé étrangère pour le champ *rwy_num* de la couche
- RunwayThreshold*.
- type : Chaine (0.0). Type comprenant :
    - VASI
    - PAPI Left
    - PAPI Right
    - Space Shuttle PAPI
    - Tri-colour VASI
    - o Wig-Wag lights
- true_heading_deg : Réel (6.2)
- visual_glide_deg : Réel (4.2)

1.q ATCFreq (None)

Définie une réquence ATC d'nu aéroport. Notez que cette couche n'a pas de géométrie.

Champs :

- apt_icao : Chaine (4.0)
- atc_type : Chaine (4.0). Type de la fréquence parmi (dérivé du numéro du type d'enregistrement) :
    - ATIS : AWOS (*Automatic Weather Observation System*), ASOS (*Automatic Surface Observation System*) ou ATIS (*Automated Terminal Information System*)
        - CTAF : Unicom ou CTAF (USA), radio (UK)
        - CLD : Clearance delivery (CLD)
        - GND : Sol
        - TWR : Tour
        - APP : Approche
        - DEP : Départ
- freq_name : Chaine (0.0). Nom de la fréquence ATC. C'est souvent une abréviation (tel que GND pour "*Ground*").
- freq_mhz : Réel (7.3). Fréquence en MHz.

D Aides à la navigation (nav.dat)

Ce fichier contient la description de divers phare d'aides à la navigation.

Les couches suivantes sont retournées :

- ILS (Point)
- VOR (Point)
- NDB (Point)
- GS (Point)
- Marker (Point)
- DME (Point)
- DMEILS (Point)

1 ILS (Point)

Localiser qui est une partie d'un ILS complet, ou un *localiser* indépendant (LOC) également inclut un LDA (*Landing Directional Aid*) ou un SDF (*Simplified Directional Facility*).

Champs :

- navaid_id : chaine (4.0). Identification du *nav-aid*. *NON* unique.
- apt_icao : chaine (4.0). Clé étrangère du champ *apt_icao* de la couche *RunwayThreshold*
- rwy_num : chaine (3.0). Clé étrangère du champ *rwy_num* de la couche *RunwayThreshold*.
- subtype : chaine (10.0). Sous-type dont :
    - ILS-cat-I
    - ILS-cat-II
    - ILS-cat-III
    - LOC
    - LDA
    - SDF
    - IGS
    - LDA-GS
- elevation_m : réel (8.2). Élévation d'un *nav-aid* en mètres.
- freq_mhz : réel (7.3). Fréquence d'un *nav-aid* en MHz.
- range_km : réel (7.3). Largeur d'un *nav-aid* en km.
- true_heading_deg : réel (6.2). En-tête réel du *localiser* en degré.

2 VOR (Point)

Navaid du type *VOR*, *VORTAC* ou *VOR-DME*.

Champs :

- navaid_id : chaine (4.0). Identification de nav-aid. *NON* unique.
- navaid_name : chaine (0.0)
- subtype : chaine (10.0). Fonction *VOR*, *VORTAC* ou *VOR-DME*.
- elevation_m : réel (8.2)
- freq_mhz : réel (7.3)
- range_km : réel (7.3)
- slaved_variation_deg : réel (6.2). Indique la variation du *cylindre* d'un *VOR/VORTAC* en degré.

3 NDB (Point)

Champs :

- navaid_id : chaine (4.0). Identification de nav-aid. *NON* unique.
- navaid_name : chaine (0.0)
- subtype : chaine (10.0). Fonction de NDB, LOM, NDB-DME.
- elevation_m : réel (8.2)
- freq_khz : réel (7.3). Fréquence en kHz
- range_km : réel (7.3)

4 GS - Glideslope (Point)

Glideslope nav-aid.

Champs :

- navaid_id : chaine (4.0). Identification du nav-aid. *NON* unique.
- apt_icao : chaine (4.0). Clé étrangère du champ *apt_icao* de la couche *RunwayThreshold*.
- rwy_num : chaine (3.0). Clé étrangère du champ *rwy_num* de la couche *RunwayThreshold*.
- elevation_m : réel (8.2)
- freq_mhz : réel (7.3)
- range_km : réel (7.3)
- true_heading_deg : réel (6.2). En-tête réel du *glideslope* en degré.
- glide_slope : réel (6.2). Angle du *glide-slope* en degré (typiquement 3 degrés)

5 Marker - ILS marker beacons. (Point)

Nav-aids de type *Outer Marker* (OM), *Middle Marker* (MM) ou *Inner Marker* (IM).

Champs :

- apt_icao : chaine (4.0). Clé étrangère du champ *apt_icao* de la couche *RunwayThreshold*.
- rwy_num : chaine (3.0). Clé étrangère du champ *rwy_num* de la couche *RunwayThreshold*.
- subtype : chaine (10.0). Fonction de OM, MM ou IM.
- elevation_m : réel (8.2)
- true_heading_deg : réel (6.2). En-tête réel du *glideslope* en degré.

6 DME (Point)

DME, incluant l'élément DME d'un VORTAC, VOR-DME ou NDB-DME.

Champs :

- navaid_id : chaine (4.0). Identification de nav-aid. *NON* unique.
- navaid_name : chaine (0.0)
- subtype : chaine (10.0). Fonction de VORTAC, VOR-DME, TACAN ou NDB-DME
- elevation_m : réel (8.2)
- freq_mhz : réel (7.3)
- range_km : réel (7.3)
- bias_km : réel (6.2). Ce biais doit être soustrait à la distance calculé au DME pour donner la lecture du cockpit désirée.

7 DMEILS (Point)

Élément DME d'un ILS.

Champs :

- navaid_id : chaine (4.0). Identification du nav-aid. *NON* unique.
- apt_icao : chaine (4.0). Clé étrangère pour le champ *apt_icao* de la couche *RunwayThreshold*.
- rwy_num : chaine (3.0). Clé étrangère pour le champ *rwy_num* de la couche *RunwayThreshold*.
- elevation_m : réel (8.2)
- freq_mhz : réel (7.3)
- range_km : réel (7.3)
- bias_km : réel (6.2). Ce biais doit être soustrait de la distance calculée au DME pour donner la lecture du cockpit désiré.

E Intersections IFR (fix.dat)

Ce fichier contient les intersections IFR (souvent nommé *fixes*).

La couche suivante est renvoyée :

- FIX (Point)

1 FIX (Point)

Champs :

- fix_name : Chaine (5.0). Nom de l'intersection. *NON* unique.

F Airways (awy.dat)

Ce fichier contient la description des segments de la route de vol.

Les couches suivantes sont retournées :

- AirwaySegment (chaine ligne)
- AirwayIntersection (point)

1 AirwaySegment (Line String)

Champs :

- segment_name : chaine (0.0)
- point1_name : chaine (0.0) : Nom de l'intersection ou nav-aid au début de ce segment
- point2_name : chaine (0.0) : Nom de l'intersection ou nav-aid au début de ce segment
- is_high : entier (1.0) : Définie à 1 si c'est une route aérienne "High"
- base_FL : entier (3.0) : Niveau de vol (en centaine de pied) de la base de la route de vol.
- top_FL : entier (3.0) : Niveau de vol (en centaine de pied) du haut de la route de vol.

2 AirwayIntersection (Point)

Champ :

- name : Chaine (0.0) : Nom de l'intersection ou nav-aid

Chapitre CXLII Python

Chapitre CXLIII Introduction

A Installation

Il existe plusieurs possibilités pour installer des extensions Python. Celles proposaient ici sont indépendantes du système et permettent une certaine indépendances avec celui-ci.

Pour Windows installez Python pour Windows

1 Système

```
$ sudo easy_install GDAL
```

Ou (mais lié à un système Debian ou Ubuntu) :

```
$ sudo apt-get install python-gdal
```

2 Environnement virtuel

Installez les paquetages libgdal, exemple pour Ubuntu/Debian :

```
$ sudo apt-get install libgdal libdal1-dev
```

Installez l'environnement virtuel :

```
$ sudo easy_install virtualenv
$ virtualenv --no-site-packages env
```

L'option `--no-site-packages` permet de ne pas mélanger le répertoire site-package du système de celui de l'environnement virtuel.

Puis activez (ie entrez dans l'environnement virtuel) :

```
$ source bin/activate
```

La console change :

```
(env)yves@helios:~/Documents/Geomatique/OSGeo/Projects/gdal/env$
```

Vous pouvez maintenant installer les extensions nécessaire :

```
(env)$ cd env
(env)$ gdal-config --version
(env)$ bin/easy_install GDAL
```

Un problème ? Notez la version de GDAL (1.8.0 pour mon cas) puis essayer cette deuxième méthode :

```
(env)$ pip install --no-install "GDAL>=1.7"
```

Vous obtenez encore :

```
Could not run gdal-config!!!!
Successfully downloaded GDAL
```

Puis lancer ces commandes :

```
(env)$ rm -f build/gdal/setup.cfg
(env)$ cd build/gdal
(env)$ python setup.py build_ext --gdal-config=gdal-config \
    --library-dirs=/usr/lib \
    --libraries=gdal1.8.0 \
    --include-dirs=/usr/include/gdal \
install
```

Testez l'installation :

```
(env)$ bin/python
```

Puis :

```
>>> from osgeo import gdal
```

Pour en sortir :

```
(env)$ deactivate
```

B API

Utilisation de Doxygen pour générer l'API : http://softlibre.free.fr/gdal/osgeo.ogr.html

```
$ pydoc -w ogr

$ epidoc ou sphinx
```

C Utiliser la bibliothèque Python

Si les étapes précédentes ont été réussi vous devez avoir la possibilité d'utiliser la bibliothèque GDAL-OGR en Python et avoir l'API au format HTML.

1 La classe GDAL

Pour utiliser GDAL :

```
>>> try:
...     from osgeo import gdal
... except ImportError:
...     import gdal
```

Explication : le bind Python de GDAL a évolué et a été incorporé dans une extension plus large nommé *osgeo*. La méthode ci-dessus permet d'être indépendante des versions de GDAL. Ainsi si la première partie (après le `try:`) échoue la deuxième partie est lancé.

GDAL propose plusieurs classes d'objet nécessaire à la manipulation de raster : les classes *band*, *DataSet*, *Driver*, *RasterAttributeTable* et *ColorTable*.

La procédure pour manipuler des données raster est celle-ci :

- chargement des drivers et activation de celui qui est nécessaire (utilisation de la classe Driver) ;
- chargement du jeu de données via l'objet driver créé (la méthode Open() renvoie un objet DataSet) ;
- chargement de la bande nécessaire (la méthode GetRasterBand() renvoie un objet Band) ;
- transformation des données en Array pour manipulation des données ;
- transformation de l'array en données raster ;
- sauvegarde du raster
- destruction des objets utilisés : dataset, bande, driver, etc.

2 La classe OGR

Les classes OGR ont évoluées depuis quelques versions. Elles font désormais partie de la classe *osgeo*. Pour garder une compatibilité avec les anciennes versions il est conseillé d'importer la classe *osgeo* et de le placer dans une méthode *try: except:* :

Pour importer les classes OGR :

```
>>> try:
...     from osgeo import ogr
...     from osgeo import osr
... except ImportError:
...     import ogr
...     import osr
```

La classe *osr* permet la gestion des systèmes de projection alors que la classe *ogr* permet de lire les fichiers vectoriels.

Chapitre CXLIV Vecteurs

A Lire un fichier vecteur

La lecture d'un fichier vecteur se fait en plusieurs étape :

- ouverture du fichier avec le pilote, on obtient l'objet initiale ;
- lecture de la couche de cet objet; on obtient l'objet layer ;
- lecture des objets (feature) de l'objet layer, on obtient l'objet feature.

Pour chaque objet on a un ensemble de méthodes liées à la classe à laquelle il appartient. Les méthodes peuvent être listées grâce à la méthode `dir(objet)`. Exemple `dir(ogr)` mais vous pouvez aussi regarder la doc de l'API Python.

Un fichier vecteur est lu grâce à la méthode *Open()* de la classe *OGRSFDriverRegistrar*, exemple :

```
ds_ref = ogr.Open("/url/to/file.shp", update = 0)
```

Nous obtenons un objet initial, quelles sont les méthodes que nous pouvons lui appliquer ? Facile : `dir(ds_ref)` ou http://wiki.gloobe.org/gdal/classosgeo_1_1ogr_1_1DataSource.html

Vous devez ensuite récupérer la ou les couche(s) grâce à *GetLayer()*, pour un shape, le nom de la couche est la même que le nom du fichier sans l'extension sinon vous devez utiliser le numéro de la couche :

```
Layer_ref = ds_ref.GetLayer()
```

La variable *Layer_ref* contient maintenant l'objet *layer*. Si vous ne connaissez pas le numéro de la couche, vous pouvez utiliser la méthode *GetLayerByName()* pour récupérer l'objet layer.

Vous pouvez ainsi accéder aux informations de cette couche :

```
Layer_ref.GetName() #Ne devrait pas être possible, GetName() n'existe
pas pour l'objet OGRLayer !
Layer_ref.GetFeatureCount()
```

```
feature = Layer_ref.GetFeature(1) # Récupère l'objet (feature) numéro
1
```

Vous pouvez trouvez plus d'information dans la classe *OGRDataSource*.

Lorsque vous avez l'objet *feature*, vous pouvez gérer vos objets géométriques avec la classe *Feature*.

```
import os, sys, string,  shutil
from osgeo import ogr

#Nom et géométrie d'un shapefile
ds_ref = ogr.Open('D:/monShapeFile.shp', update = 0)
Layer_ref = ds_ref.GetLayer()
print "Layer name : " + Layer_ref.GetName()
print "Number of features : " + str(Layer_ref.GetFeatureCount())

#Sélection sur un shapefile
query='SURFACE > 1000'
Layer_ref.SetAttributeFilter(query)

# Export de la sélection
driver = ogr.GetDriverByName('ESRI Shapefile')
ds_dest=driver.CreateDataSource('D:/maSelection.shp')
ds_dest.CopyLayer (Layer_ref ,  'maSelection')
ds_dest.Destroy()

ds_ref.Destroy()
```

B Écrire un fichier vecteur

```
# Create a Shapefile
output="d:/temp/testogr.shp"
driver = ogr.GetDriverByName('ESRI Shapefile')
if os.path.exists(output):
    driver.DeleteDataSource(output)
ds = driver.CreateDataSource(output)
layer = ds.CreateLayer(output, geom_type=ogr.wkbPolygon)
```

1 Création de shapefile à partir de coordonnées

Pour créer des objets géométriques à partir de coordonnées une possibilité est de passer par Wkt ([http://en.wikipedia.org/wiki/Well-known_text Well-known text]). Il s'agit d'une simple chaîne de caractère qui permet de définir le type d'objet et les coordonnées associées.

Exemple de Wkt :

```
wkt_point = 'POINT(6 10)'
wkt_ligne = 'LINESTRING(3 4,10 50,20 25)'
wkt_polygone = 'POLYGON((1 1,5 1,5 5,1 5,1 1),(2 2, 3 2, 3 3, 2 3,2
2))'
```

Vous pouvez alors créer un objet *Geometry* en utilisant la méthode *CreateGeometryFromWkt*.

Exemple : création d'un shapefile de points à partir des coordonnées x y

```
#Création d'un shapefile de points
driver = ogr.GetDriverByName('ESRI Shapefile')
ds = driver.CreateDataSource('D:/monShapefile.shp')
layer = ds.CreateLayer('monShapefile', geom_type=ogr.wkbPoint)

#
feature_def=layer.GetLayerDefn()
f = ogr.Feature(feature_def)

# Ajout du point
x = 731065
y = 2368493
wkt = 'POINT(%f %f)' % (x, y)
p = ogr.CreateGeometryFromWkt(wkt)
f.SetGeometryDirectly(p)
layer.CreateFeature(f)

f.Destroy()
```

C Projections

Exemple : définir une projection (Lambert 2 étendu) à un shapefile :

```
# Crée le système de référence spatiale
lambert2e = osr.SpatialReference()
lambert2e.ImportFromEPSG(27572)

#Crée le fichier .proj
lambert2e.MorphToESRI()
wkt_proj = lambert2e.ExportToWkt()
prj_file = open('D:/monShapefile.prj', 'w')
prj_file.write(wkt_proj)
prj_file.close()
```

Les manipulations de systèmes de projections se font sur les objets *Geometry*. On peut soit utiliser la méthode *TransformTo* soit *Transform*.

- TransformTo suppose qu'un système de coordonnées a été défini pour l'objet *Geometry* et qu'un objet *SpatialReference* a été défini comme système de coordonnées d'arrivé.
- *Transform* suppose qu'un objet *CoordinateTransformation* ait été défini. Il n'est pas indispensable d'avoir défini le système de coordonnées de l'objet *Geometry* traité, il sera supposé que ce système correspond à celui de départ de l'objet *CoordinateTransformation*.

Il est conseillé d'utiliser plutôt *Transform* et donc de définir au préalable un objet *CoordinateTransformation* pour des modifications sur un grand nombre d'objets.

1 Comment Reprojeter des données

On créé deux objets Références Spatiales et pour chacun on importe les données de

reprojection via un import à partir d'un code EPSG. Puis on créé un objet Géométrie à partir d'un WKT. On lui assigne un projection, puis on reprojète l'objet.

```
to_srs = osr.SpatialReference()
to_srs.ImportFromEPSG(4326)
from_srs = osr.SpatialReference()
from_srs.ImportFromEPSG(900913)

wkt = 'POINT(%f %f)' % (x, y)

pt = ogr.CreateGeometryFromWkt(wkt)
pt.AssignSpatialReference(from_srs)
pt.TransformTo(to_srs)
geom = pt.GetX(),pt.GetY()
return pt
```

D Requêtes spatiales

1 Par bbox

2 Par objet géométrique

3 Par requête SQL

Chapitre CXLV Raster

A Lire un fichier Raster

1 Les drivers

Vous devez charger les drivers pour les utiliser. Pour charger les drivers en une fois (pour lire un raster seulement, pas pour écrire), utiliser :

```
gdal.AllRegister()
```

Puis créez votre objet driver :

```
driver = gdal.GetDriverByName('SRTMHGT')
```

enfin enregistrez le :

```
driver.register()
```

2 Lire un fichier raster

Vous pouvez maintenant lire un fichier raster sous forme de jeu de données (dataset) :

```
file = '~/Local/Data/Raster/N43E004.hgt'
ds = gdal.Open(file, GA_ReadOnly)
```

La méthode Open() prend deux paramètres : le chemin du fichier et la méthode de lecture. Vous avez deux constantes possibles pour la méthode de lecture :

```
GA_ReadOnly = 0
GA_Update = 1
```

Si python vous renvoie un message d'erreur sur la constante *GA_ReadOnly*, vous pouvez

la remplacer par sa valeur (0 donc).

Lorsque le jeu de données est chargé on test si son ouverture ne pose pas de problème :

```
if ds is None:
    print 'impossible d ouvrir '+file
    sys.exit(1)
```

Voici quelques méthodes définies pour récupérer de l'information sur vos données :

```
ds.GetProjection() # obtenir la projection
```

En plus des méthodes définie dans l'API il existe trois propriétés :

```
ds.RasterXSize
ds.RasterYSize
ds.RasterCount
```

3 Bande

Pour travailler sur les pixels, nous devons obtenir la bande :

```
band = ds.GetRasterBand(1)
```

Puis récupérons les données dans un tableau :

```
data = band.ReadAsArray(xOffset, yOffset, 1, 1)
```

1,1 est la taille de la cellule que nous voulons récupérer.

xOffset et yOffset sont obtenu en calculant la distance en pixel entre le bord en haut à gauche et le point pour chaque axes (ordonnés et abscisses). Nous connaissons les coordonnées du point haut gauche, la taille d'une cellule en pixel. Nous avons donc :

```
xOffset (en pixel) = (originX - X) (en coordonnées)
1 pixel            = pixelWidth (en coordonnées)
-------------------------------------------------
xOffset = (originX - X) (en coordonnées) / pixelWidth
```

Même chose pour les ordonnées !

Si notre data set est la couche N43E004.hgt (format SRTM), gdalinfo nous donne ceci :

```
$ gdalinfo N43E004.hgt
Driver: SRTMHGT/SRTMHGT File Format
Files: N43E004.hgt
Size is 1201, 1201
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
```

```
        TOWGS84[0,0,0,0,0,0,0],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9108"]],
    AXIS["Lat",NORTH],
    AXIS["Long",EAST],
    AUTHORITY["EPSG","4326"]]
Origin = (3.999583333333333,44.000416666666666)
Pixel Size = (0.000833333333333,-0.000833333333333)
Corner Coordinates:
Upper Left  (   3.9995833,  44.0004167) (  3d59'58.50"E, 44d 0'1.50"N)
Lower Left  (   3.9995833,  42.9995833) (  3d59'58.50"E,
42d59'58.50"N)
Upper Right (   5.0004167,  44.0004167) (  5d 0'1.50"E, 44d 0'1.50"N)
Lower Right (   5.0004167,  42.9995833) (  5d 0'1.50"E, 42d59'58.50"N)
Center      (   4.5000000,  43.5000000) (  4d30'0.00"E, 43d30'0.00"N)
Band 1 Block=1201x1 Type=Int16, ColorInterp=Undefined
NoData Value=-32768
Unit Type: m
```

En python faîtes :

```
geotransform = ds.GetGeoTransform()
originX = geotransform[0]
originY = geotransform[3]
pixelWidth = geotransform[1]
pixelHeight = geotransform[5]
```

Si nous cherchons la valeur de la coordonnées (43.2, 4.2) :

```
xOffset = int((4.2 - originX) / pixelWidth)
yOffset = int((43.2 - originY) / pixelHeight)
```

Si vous avez un offset négatif, il est fort probable que vous ayez choisit un point en dehors de la zone de couverture du raster. Nous pouvons relancer la commande suivante :

```
data = band.ReadAsArray(xOffset, yOffset, 1, 1)
```

Les données (cad la variable data) est un tableau en 2 dimensions de la taille qui a été définir plus haut (1,1). Pour récupérer une valeur du tableau :

```
value = data[0,0]
```

Comment récupérer tous le raster dans le tableau à deux dimensions ? En définissant les offset à 0 et en donnant la largeur et la hauteur du raster dans la taille de la cellule à récupérer.

Le tableau de valeur est un tableau de colonne, les deux valeurs sont bien des colonnes et des lignes et non des coordonnées. De plus la première ligne et la première colonne commencent à 0 !

Note!

Quelques conseils :

Ne lisez pas un pixel à chaque fois mais récupérer les tous en une fois, puis traiter les. Ne lisez qu'un pixel à la fois si vous êtes sur d'en avoir besoin que d'un ou deux ! Malheureusement pour de gros jeux de données, cela peut poser problème ;) La solution est d'utiliser la taille des blocs ou de lire une ligne et de faire le traitement voulut, puis la ligne suivante. Ne traitez pas un pixel à la fois, mais utiliser les fonctions built-in de Python, notamment le module Numpy ou Numeric.

B Écrire un fichier Raster

Chapitre CXLVI Bibliographie

- GDAL-OGR python
- PerryGeo Tissot Indicatrix
- Graticule Hacking with OGR
- PostGIS Wiki : OGR Examples
- Bio and Geo Informatics : OGR python projection
- WorldMill - PCL - Trac
- Using th OGR Python bindings for the study on ancient monuments - Ominiverdi's blog
- http://www.gis.usu.edu/~chrisg/python/

Chapitre CXLVII FAQ

Chapitre CXLVIII FAQ Générale

A Qu'est ce qu'est GDAL ?

Le nom GDAL est habituellement utilisé pour nommer ce qui suit :

- le projet de bibliothèque de traduction pour les raster (GDAL) et les formats de données vectoriels (OGR), dans ce cas GDAL = GDAL + OGR
- la bibliothèque de traduction pour les formats de données géospatiales.

Quelques notes historiques peuvent être trouvées ici.

B Que signifie GDAL ?

GDAL - Geospatial Data Abstraction Library, Bibliothèque d'Asbtraction de Données Géospatiales.

Il est parfois prononcé *goo-doll* (un peu comme goo-gle), bien que d'autres le prononce *gee-doll* en anglais et G-dal en français.

C À quoi sert OGR ?

L'arborescence de GDAL/OGR contient les sources pour une bibliothèque entré/sortie vectorielle inspiré par les Simple Features de l'OpenGIS. En théorie il est séparé de GDAL mais réside aujourd'hui dans la même arborescence de la source et est en quelque sorte empêtré. Vous pouvez trouver plus d'information sur http://www.gdal.org/ogr/. C'est un des objectifs de réunir proprement OGR dans GDAL dans le futur. GDAL sera alors une bibliothèque raster et vecteur.

D Que signifie OGR ?

OGR signifiait *OpenGIS Simple Features Reference Implementation*. Cependant, puisqu'OGR ne suit pas complètement la spécification Simple Feature d'OpenGIS et n'est pas approuvé comme une implémentation de référence de la spécification le nom a été changeait en *OGR Simple Features Library*. Le terme d'OGR dans le nom est purement historique. OGR est également le préfixe utilisé partout dans le code source pour les noms des classes, des fichiers dans la bibliothèque, etc.

E Quand le projet GDAL a t-il démarré ?

Fin 1998, Frank Warmerdam a commencé à travailler comme consultant sur la bibliothèque GDAL/OGR.

F Est ce que GDAL/OGR est un logiciel propriétaire ?

Non ! GDAL/OGR est un Logiciel Libre et Open Source.

G Quelle licence utilise GDAL/OGR ?

La bibliothèque GDAL/OGR est diffusé sous les termes de la Licence X11 / MIT.

Elle a pour but de vous donner la permission de faire ce que vous voulez avec le code source de GDAL : télécharger, modifier, redistribuer comme vous le souhaitez, incluant la compilation en un logiciel commercial et propriétaire, aucune permission n'est nécessaire de la part de Frank Warmerdam, la Fondation OSGeo ou quiconque.

Quelques portions de GDAL est sous des termes un peu différents. Par exemple les termes de la licence des bibliothèques libpng, libjpeg, libtiff, et libgeotiff peuvent varier sensiblement mais pas de manière significative. D'autres bibliothèques qui peuvent être utilisé par GDAL sont sous des licences radicalement différentes.

```
Copyright (c) 2000, Frank Warmerdam

Permission is hereby granted, free of charge, to any person obtaining
a
copy of this software and associated documentation files (the
"Software"),
to deal in the Software without restriction, including without
limitation
the rights to use, copy, modify, merge, publish, distribute,
sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

H Sur quels systèmes fonctionne GDAL-OGR ?

Vous pouvez utiliser GDAL/OGR sur tous les Unix moderne : Linux, Solaris, Mac OS X ; et les versions les plus récente de Microsoft Windows (NT/2000/XP/Vista/CE). À la fois les architectures 32-bit et 64-bit sont gérés.

Si vous avez utilisé GDAL/OGR sur un système ou architecture qui n'est pas listé au-dessus, n'hésitez pas à nous le dire.

I Y a t-il des interfaces graphiques pour GDAL/OGR ?

Voyez la page :ref:`gdal.logicielutilisantgdal`.

Unknown interpreted text role "ref".

J Quel compilateur puis je utiliser pour compiler GDAL/OGR ?

GDAL/OGR est écrit en C ANSI et C++. Il peut être compilé avec tous les compilateurs C/C++ modernes.

K J'ai une question. Où puis je trouver plus d'informations ?

Si vous ne trouvez pas de réponse après avoir naviguer dans cette FAQ, il y a plusieurs autres ressources disponibles :

- la documentation disponible sur les sites de site de GDAL et OGR ;
- les pages Wiki
- les listes de diffusions en anglais
- le canal IRC #gdal sur irc.freenode.net ;
- le forum geoLibre sur le portail GeoRezo.net en français.

Garder à l'esprit que la qualité des réponses est en relation à la qualité de votre question. Si vous avez besoin d'explication détaillé à ce propos, vous pouvez trouver cela dans De la bonne manière de poser les questions par Eric S. Raymond.

L Quand est prévue la prochaine version ?

Lisez la page de planification.

M Comment puis je ajouter un nouveau format à gérer ?

C'est maintenant couvert par le Tutorial de Développement de Pilote pour GDAL (GDAL Driver Implementation Tutorial) et le Tutorial de Développement de Pilote pour OGR (OGR Driver Implementation Tutorial), tout deux en anglais.

Chapitre CXLIX FAQ installation et compilation

A Où est ce que je peux trouver une version de développement de GDAL ?

Vous pouvez la récupérer directement du dépôt SVN. Visitez la page Wiki Downloading GDAL/OGR Source pour des instructions détaillées.

B Puis je avoir un fichier de projet pour MS Visual Studio pour GDAL ?

Les développeurs de GDAL trouve plus pratique de compiler avec `makefiles` et l'utilitaire Visual Studio NMAKE. Maintenir un ensemble parallèle de fichier projet pour GDAL implique trop de travail, il n'y a donc pas de fichiers de projet complet directement disponible des mainteneurs.

Il existe des fichiers projet très simple disponible depuis GDAL/OGR 1.4.0 qui invoque juste les makefiles pour la compilation, mais permet le débugage plus pratique. Ce sont les fichiers *makegdal71.vcproj* et *makegdal80.vcproj* dans le répertoire racine de GDAL. Lisez la page *Using Makefile Projects <http://trac.osgeo.org/gdal/wiki/MakeFileProjects>* pour plus de détails.

Occasionnellement d'autres utilisateurs prépare des fichiers projets complets, et vous pouvez y avoir accès en le demandant sur la liste gdal-dev list. Cependant, je vous recommande fortement d'utiliser le système compilé basé sur `NMAKE`. Avec le mode débugage activé vous pouvez encore débuguer dans GDAL avec Visual Studio.

C Puis je compiler GDAL avec MS Visual C++ 2008 Édition Express ?

Oui, depuis au moins GDAL/OGR 1.5 ce qui devrait être simple. Avant de procéder à la compilation normale basée sur NMAKE faite simplement en sorte que la macro MSVC_VER vers le haut du fichier *GDAL/nmake.opt* soit changé en 1500 pour tenir compte de la version du compilateur (Visual C++ 9.0). Cela va modifier la compilation

pour sauter l'interface VB6 qui dépend de composants ATL non disponibles dans l'édition express.

Téléchargement de Microsoft Édition Express

D Puis je compiler GDAL avec MS Visual C++ 2005 Express Edition ?

Oui, vous pouvez. Il est également possible d'utiliser les bibliothèques GDAL dans une application développée en utilisant Microsoft Visual C++ 2005 Express Edition.

- Téléchargez et installez Visual C++ 2005 Express Edition. Suivez les instructions présentées sur ce site.
- Téléchargez et installez Microsoft Platform SDK. Encore, suivez les instructions soigneusement sans omettre une étape.
- ajouter les chemins suivants pour inclure les fichiers dans les paramètres de l'IDE de Visual C++. Faites le de la même manière présenté dans l'étape 3 du site ci-dessus.
  C:\Program Files\Microsoft Platform SDK\Include\atl C:\Program Files\Microsoft Platform SDK\Include\mfc
- Puisque vous voulez compiler GDAL à partir de la ligne de commande en utilisant l'outil `nmake`, vous avez besoin également de définir ou de mettre à jour les variables d'environnement *INCLUDE* et *LIB* manuellement. Vous pouvez le faire de deux manières :
  - en utilisant l'appliet système disponible dans le panneau de contrôle ;
  - en éditant le script *vsvars32.bat* localisé dans *C:Program FilesMicrosoft Visual Studio 8Common7Toolsvsvars32.bat*

Ces variables doivent avoir les valeurs assignées suivantes :

```
INCLUDE=C:\\Program Files\\Microsoft Visual Studio 8\\VC\\Include;
    C:\\Program Files\\Microsoft Platform SDK\\Include;
    C:\\Program Files\\Microsoft Platform SDK\\Include\\mfc;
    C:\\Program Files\\Microsoft Platform SDK\\Include\\atl;%INCLUDE%

LIB=C:\\Program Files\\Microsoft Visual Studio 8\\VC\\Lib;
    C:\\Program Files\\Microsoft Visual Studio 8\\SDK\\v2.0\\lib;
    C:\\Program Files\\Microsoft Platform SDK\\lib;%LIB%
```

Note!

> *Si vous avez édité les variables système LIB et INCLUDE en utilisant l'applet système, chaque console (cmd.exe) sera proprement définie. Mais si vous les avez édité par le script vsvars32.bat, vous devrez lancé ce script avant chaque compilation.</note>*

- Patchez l'en-tête *atlwin.h* : à la ligne 1725 ajoutez la déclaration *int i;* afin que cela ressemble à ceci :

```
BOOL SetChainEntry(DWORD dwChainID, CMessageMap* pObject, DWORD
dwMsgMapID = 0)
{
    int i;
```

```
    // first search for an existing entry

    for(i = 0; i < m_aChainEntry.GetSize(); i++)
```

- Patchez l'en-tête *atlbase.h* : à la ligne 287, commentez les fonctions *AllocStdCallThunk?* et *FreeStdCallThunk?* et ajoutez les macros de remplacement :

```
    /*************************************************
    PVOID __stdcall __AllocStdCallThunk(VOID);
    VOID __stdcall __FreeStdCallThunk(PVOID);

    #define AllocStdCallThunk() __AllocStdCallThunk()
    #define FreeStdCallThunk(p) __FreeStdCallThunk(p)

    #pragma comment(lib, "atlthunk.lib")
    *************************************************/

    /* NEW MACROS */
    #define AllocStdCallThunk()
    HeapAlloc(GetProcessHeap(),0,sizeof(_stdcallthunk))
    #define FreeStdCallThunk(p) HeapFree(GetProcessHeap(), 0, p)
```

- Compilez GDAL :
  - Ouvrez une console Windows (Démarrez -> Éxécutez -> cmd.exe -> OK)
  - si vous avez édité le script *vsvars32.bat*, vous devez le lancer en utilisant le chemin complet : `C:\> "C:\\Program Files\\Microsoft Visual Studio 8\\Common7\\Tools\\vsvars32.bat"`
- Définir l'environnement pour utiliser les outiles Microsoft Visual Studio 2005 x86 :
  - Allez dans le répertorie racine des sources de GDAL, par exemple `C:\> cd work\gdal`
  - Lancez `nmake` pour compiler `C:\work\gdal> nmake /f makefile.vc`
  - Si aucune erreur n'est apparue, après quelques minutes vous devez voir les bibliothèques GDAL dans *C:workgdal*.

Maintenant vous pouvez utiliser ces bibliothèques dans vos applications développées en utilisant Visual C++ 2005 Express Edition.

E Puis je compiler GDAL avec Cygwin ou MinGW ?

GDAL peut être compilé avec Cygwin en utilisant la méthodologie de compilation du style type-Unix. Il est également possible de compiler avec MinGW <http://www.mingw.org/>`_ et MSYS bien qu'il peut y avoir des complications. Ce qui suit devrait fonctionner :

```
./configure --prefix=$PATH_TO_MINGW_ROOT --host=mingw32 \
    --without-libtool --without-python $YOUR_CONFIG_OPTIONS
```

Utiliser des bibliothèques win32 externe sera souvent problématique avec l'un de ses environnements - cela nécessitera au moins un développement du fichier *GDALmake.opt*.

Comment compiler le bindings Python (NG) :

```
cd swig\python
python setup.py build -c mingw32
cp build\lib.win32-2.5\* c:\python25\lib\site-packages\
```

(certains détails devront être ajusté).

Comment compiler le bindings Perl :

```
cd swig\perl
perl Makefile.PL
make.bat
make.bat install
```

(Il peut être nécessaire de compiler Perl avec MinGW)

Si vous avez `swig`, le bindings peut être régénéré dans le prompt de MSYS par la commande `make generate`.

F Puis je compiler GDAL avec Borland C ou d'autres compilateurs C ?

Ce ne sont pas des compilateurs gérés pour GDAL ; Cependant GDAL est assez générique, si vous êtes prêt à prendre en charge la tache de compiler un *makefile* approprié ou un ficher de projet cela doit être possible. vous trouverez la plupart des problèmes de compatibilité dans le fichier gdal/port/cpl_port.h et vous devrez préparer un fichier gdal/port/cpl_config.h approprié à votre plateforme. Utiliser le fichier cpl_config.h.vc comme guide peut être utile.

G Pourquoi Visual C++ 8.0 plante avec une erreur C2894 dans wspiapi.h lors de la compilation de GDAL ...

Voici le message d'erreur complet de ce problème :

```
C:\Program Files\Microsoft Visual Studio
8\VC\PlatformSDK\include\wspiapi.h(44) :
error C2894: templates cannot be declared to have 'C' linkage
```

C'est un bug connus dans l'en-tête *wspiapi.h*. Une solution possible est de patcher manuellement le fichier *curl.h* en replaçant les lignes 153 - 154 :

```
#include <winsock2.h>
#include <ws2tcpip.h>
```

par les lignes suivantes :

```
#ifdef __cplusplus
}
#endif
#include <winsock2.h>
#include <ws2tcpip.h>
#ifdef __cplusplus
```

```
extern "C" {
#endif
```

Ce problème arrive avec libcurl < 7.17.1. peut être qu'une version plus récente de `libcurl` inclura ce correctif.

H Comment puis je ajouter des LDFLAGS particulier avec GDAL < 1.5 ?

Exportez la variable *LNK_FLAGS* avec votre contenu *LDFLAGS* habituel :

```
export LNK_FLAGS=-Wl,-rpath=/foo/lib -l/foo/lib
```

I J'ai des soucis lors de la compilation avec des bibliothèques externes, que puis je faire ?

Il y a des astuces et suggestions pour la compilation de GDAL pour différentes gestions des bibliothèques externes dans le sujet Astuces.

Chapitre CL F.A.Q. Raster

A Pourquoi gdalwarp ou gdal_merge n'écrit pas la plupart des formats ?

GDAL gère beaucoup de formats raster en lecture, mais beaucoup moins de formats en écriture. Parmi ceux gérés en écriture la plupart ne sont gérés seulement en mode *create copy*. Essentiellement cela signifie qu'ils doivent être écrits séquentiellement à partir d'une copie en entrée fournie de l'image à écrire. Les programmes tels que `gdal_merge.py` ou `gdalwarp` qui écrit des morceaux d'image non séquentiellement ne peuvent pas écrire facilement vers ces formats écrit séquentiellement. D'une manière générale les formats qui sont compressés, tel que PNG, JPEG et GIF sont à écriture séquentielle. Aussi certains formats nécessitent que des informations telles que le système de coordonnées, et la table de couleur soient connues au moment de la création et donc ceux-ci sont aussi des formats à écriture séquentielle.

Quand vous rencontrez ce problème il est généralement pas une bonne idée d'écrite d'abord le résultat au format GeoTIFF puis de traduire vers le format cible.

Pour déterminer quels formats gèrent quelles possibilités, utilisez l'option *--formats* avec à peu près toutes les commandes GDAL. Chaque pilote inclura soit r (lecture seule), rx (lecture et écriture séquentielle) ou rw+ (lecture, écriture séquentielle ou aléatoire).

B Comment améliorer les performances de gdalwarp ?

Brièvement : utilisez la mémoire déformée et configurez le paramètre *cachemax*. Par exemple `gdalwarp --config GDAL_CACHEMAX 500 -wm 500` utilise 500MB de RAM pour lire/écrire le cache, et 500 MB de RAM pour les buffers de travail pendant la déformation.

Pour plus de détails, lisez UserDocs/GdalWarp.

C Comment convertir un raster en une couche de polygones ?

TBD

D Comment puis je créer un raster blanc basé sur l'étendue de fichiers vecteurs pour utiliser avec gdal_rasterize ?

Aujourd'hui la commande `gdal_rasterize` ne peut pas créer de raster blanc basé sur une source de données vecteur bien que cette fonctionnalité ait été considérée ([[http://trac.osgeo.org/gdal/ticket/1599|Ticket #1599]]). Jusqu'à ce que cette fonctionnalité soit ajouté, vous devrez trouver un moyen de générer un raster qui correspond à l'étendue de votre couche vectorielle. Vous pouvez alors utiliser ce raster blanc à partir de votre couche vectorielle.

Une approche est d'utiliser `gdal_translate` pour découper et aplanir un fichier raster existant qui couvre votre zone d'intérêt. Lisez cet exemple de syntaxe.

Une autre approche est d'utiliser un des langages de GDAL pour créer un raster blanc à partir de rien. La part la plus difficile est la compréhension de la syntaxe de *GeoTransform*. Cet extrait en Python montre comment lire un shapefile et sortir un tiff qui correspond à l'étendue du shapefile.

Note!

Vous devez modifier la variable px basée sur la résolution désirée et la dimension du raster.

```
#!/usr/bin/env python

from osgeo import gdal
from osgeo import osr
from osgeo import ogr
import numpy

shp = 'test.shp'
tiff = 'test.tif'
px = .001
tiff_width = 7850
tiff_height = 3500

# Import vector shapefile
vector = ogr.GetDriverByName('ESRI Shapefile')
src_ds = vector.Open(shp)
src_lyr = src_ds.GetLayerByIndex(index=0)
src_extent = src_lyr.GetExtent()

# Create new raster layer with 4 bands
raster = gdal.GetDriverByName('GTiff')
dst_ds = raster.Create( tiff, tiff_width, tiff_height, 4,
gdal.GDT_Byte)

# Create raster GeoTransform based on upper left corner and pixel
resolution
raster_transform = [src_extent[0], px, 0.0, src_extent[3], 0.0, -px]
dst_ds.SetGeoTransform( raster_transform )

# Get projection of shapefile and assigned to raster
srs = osr.SpatialReference()
srs.ImportFromWkt(src_lyr.GetSpatialRef().__str__())
dst_ds.SetProjection( srs.ExportToWkt() )
```

```
# Create blank raster with fully opaque alpha band
zeros = numpy.zeros( (tiff_height, tiff_width), numpy.uint8 )
dst_ds.GetRasterBand(1).WriteArray( zeros )
dst_ds.GetRasterBand(2).WriteArray( zeros )
dst_ds.GetRasterBand(3).WriteArray( zeros )
opaque = numpy.ones((tiff_height,tiff_width), numpy.uint8 )*255
dst_ds.GetRasterBand(4).WriteArray( opaque )
```

E Puis je utiliser gdal_rasterize pour générer des polygones "non solides" ?

Lisez la page How gdal_rasterize works dans les archives de gdal-dev.

Comme Chris Barker le suggère, les possibilités de rastérisation deGDAL sont assez limitées d'un point de vue du style du rendu. D'autres outils peuvent être plus appropriés si vous voulez faire quelque chose de plus sophistiqué que rastériser des polygones dans une seule couleur.

Duplicate explicit target name: "grass".

Duplicate explicit target name: "mapserver".

Exemples d'autres outils : Quantum GIS, GRASS, MapServer, GMT, SAGA GIS.

Cependant, si votre raster gère la transparence dans la bande alpha (RGBA), alors vous pouvez utiliser `gdal_rasterize` pour "brûler" complètement les zones transparentes dans votre image avec :

```
$ gdal_rasterize -b 4 -burn 0 -where your_field=some_value -l
your_layer your_vector_file.shp your_raster
```

F Comment utiliser gdal_translate pour extraire une sous partie d'un raster ?

`Gdal_translate` a été désigné pour convertir à partir et vers divers formats raster, mais il peut aussi réaliser des opérations de géotraitement utile durant la conversion.

Si vous désirez extraire une sous partie d'un raster vous pouvez utiliser les options *-srcwin* ou *-projwin*. Dans la terminologie GDAL, ce sont des opérations de "subsetting" qui permet de sélectionner une sous fenêtre "subwindows" pour copier à partir d'un jeu de données sources dans un jeu de données de destination.

Voici un exemple de l'utilisation de `gdal_translate` sur une orthophographie NAIP au format sid pour sélectionner une petite zone qui montre l'île Blakely, WA :

```
$ gdal_translate -projwin 510286 5385025 518708 5373405 ortho_1-
1_1n_s_wa055_2006_1.sid naip_ortho_blakely_island.tif
```

Cet exemple utilise l'option *-projwin* qui accepte les coordonnées des limites dans les coordonnés projetées plutôt qu'en pixel (*-srcwin*). *-projwin* de `Gdal_translate` nécessite les coordonnées X et Y du coin en haut à gauche, les coordonnées X et Y du coin le plus à droite. L'image NAIP dans cet exemple est en NAD 83 Utm 10, pour obtenir les coordonnées des limites j'ai simplement chargé l'inde shapfile qui est fournie avec

l'image NAIP dans Quantum GIS et lu les coordonnées sur l'écran pour former mon étendue.

Note!

Aujourdh'ui le découpage d'un raster en utilisant une étendue vectorielle polygonale n'est pas gérée, mais est en discussion (lisez http://trac.osgeo.org/gdal/ticket/1599). Cependant, il est assez facile d'obtenir l'étendue d'un shapefile donné et de convertir ses coordonnées dans la forme utilisable par `gdal_translate` *sans lire manuellement l'étendue dans une autre application comme QGIS.*

Disons que vous avez un shapefile nommé *clipping_mask.shp* utiliser `ogrinfo` pour obtenir l'étendue :

- notez que l'utilisation d'une pipe (|) et de la commande `grep` est optionnelle (| *grep Extent*), mais une manière habile de limiter l'information renvoyée par `ogrinfo` pour obtenir juste ce dont vous avez besoin :

```
$ ogrinfo clipping_mask.shp -so -al | grep Extent
# which gives the extent as xMin,yMin, xMax, yMax:
Extent: (268596, 5362330) - (278396, 5376592)
# which is (xMin,yMin) - (xMax,yMax)
```

Puis copier et coller ce texte pour créer votre commande de découpe avec `gdal_translate` :

```
# -projwin's ulx uly lrx lry is equivalent to xMin, yMax, xMax, yMin
so just switch the Y coordinates
# For the above Extent that would turn into:
$ gdal_translate -projwin 268596 5376592 278396 5362330 src_dataset
dst_dataset
```

G Comment retrouver la liste des formats supportés par ma version de GDAL ?

Utilisez la commande :

```
gdalinfo --formats
```

Celle-ci vous renvoie :

```
$ gdalinfo --formats
Supported Formats:
    GRASS (ro): GRASS Database Rasters (5.7+)
    VRT (rw+): Virtual Raster
    GTiff (rw+): GeoTIFF
    NITF (rw+): National Imagery Transmission Format
    HFA (rw+): Erdas Imagine Images (.img)
    SAR_CEOS (ro): CEOS SAR Image
    CEOS (ro): CEOS Image
    ELAS (rw+): ELAS
```

```
  AIG (ro): Arc/Info Binary Grid
  AAIGrid (rw): Arc/Info ASCII Grid
  SDTS (ro): SDTS Raster
  OGDI (ro): OGDI Bridge
  DTED (rw): DTED Elevation Raster
  PNG (rw): Portable Network Graphics
  JPEG (rw): JPEG JFIF
  MEM (rw+): In Memory Raster
  JDEM (ro): Japanese DEM (.mem)
  GIF (rw): Graphics Interchange Format (.gif)
  ESAT (ro): Envisat Image Format
  FITS (rw+): Flexible Image Transport System
  BSB (ro): Maptech BSB Nautical Charts
  XPM (rw): X11 PixMap Format
  BMP (rw+): MS Windows Device Independent Bitmap
  AirSAR (ro): AirSAR Polarimetric Image
  RS2 (ro): RadarSat 2 XML Product
  PCIDSK (rw+): PCIDSK Database File
  PCRaster (rw): PCRaster Raster File
  ILWIS (rw+): ILWIS Raster Map
  SGI (ro): SGI Image File Format 1.0
  Leveller (ro): Leveller heightfield
  GMT (rw): GMT NetCDF Grid Format
  netCDF (rw): Network Common Data Format
  PNM (rw+): Portable Pixmap Format (netpbm)
  DOQ1 (ro): USGS DOQ (Old Style)
  DOQ2 (ro): USGS DOQ (New Style)
  ENVI (rw+): ENVI .hdr Labelled
  EHdr (rw+): ESRI .hdr Labelled
  PAux (rw+): PCI .aux Labelled
  MFF (rw+): Vexcel MFF Raster
  MFF2 (rw+): Vexcel MFF2 (HKV) Raster
  FujiBAS (ro): Fuji BAS Scanner Image
  GSC (ro): GSC Geogrid
  FAST (ro): EOSAT FAST Format
  BT (rw+): VTP .bt (Binary Terrain) 1.3 Format
  LAN (ro): Erdas .LAN/.GIS
  CPG (ro): Convair PolGASP
  IDA (rw+): Image Data and Analysis
  NDF (ro): NLAPS Data Format
  DIPEx (ro): DIPEx
  ISIS2 (ro): USGS Astrogeology ISIS cube (Version 2)
  PDS (ro): NASA Planetary Data System
  JPEG2000 (rw): JPEG-2000 part 1 (ISO/IEC 15444-1)
  ECW (rw): ERMapper Compressed Wavelets
  JP2ECW (rw+): ERMapper JPEG2000
  L1B (ro): NOAA Polar Orbiter Level 1b Data Set
  FIT (rw): FIT Image
  RMF (rw+): Raster Matrix Format
  WCS (ro): OGC Web Coverage Service
  RST (rw+): Idrisi Raster A.1
  RIK (ro): Swedish Grid RIK (.rik)
  USGSDEM (rw): USGS Optional ASCII DEM (and CDED)
  GXF (ro): GeoSoft Grid Exchange Format
```

Chapitre CLI FAQ vecteur

A Comment puis je fusionner des 100e de shapefiles ?

Voici un script bash pour charger en masse un répertoire de shapefile qui ont le même schéma dans postgis. Il peut évidemment être amélioré, mais il fonctionne.

```
#!/bin/bash

# let OGR create a table from one of the files
ogr2ogr -f Postgresql PG:"host=smoke.hobu.net" -a_srs "EPSG:26915"
-nln outputlayer first_input_shape.shp -overwrite -nlt POLYGON

# delete all the data in the table we just created (but don't delete
the table)
ogrinfo PG:"host=smoke.hobu.net" -sql "delete  from outputlayer"

# loop through all of the shapefiles in the directory and load them
for i in $(ls *.shp); do
ogr2ogr -f Postgresql PG:"host=smoke.hobu.net" -a_srs " EPSG:26915"
-nln outputlayer $i -update -append -skipfailures
done
```

Note!

si vous avez une erreur d'import de PostGIS similaire à "ERROR: new row for relation "layer1" violates check constraint" vous devez essayer de substituer "-nlt GEOMETRY" par votre type de géométrie (dans l'exemple ci-dessus ce sont des POLYGON). Cela évitera des erreurs si votre shapefiles inclues des géométries de types polygones et des multipolygones, par exemple. Lisez ce message pour plus d'information : http://postgis.refractions.net/pipermail/postgis-users/2006-June/012495.html

Ces étapes de consoles Unix illustre comment batcher une fusion multiple de shapefiles en un seul shapefil en utilisant OGR. Cela suppose que vous avez téléchargé un groupe de shapefile dans un fichier .zip qui sont tous de même type de données mais nécessite

juste d'être combiné. Cela fonctionne sous Mac mais n'a pas été testé sous d'autres versions d'Unix :

```
#Make a new directory called "tmp" and a sub directory called "merged"
mkdir tmp
mkdir tmp/merged

#copy all zipped files to the "tmp" directory and then "cd" into it
cp *.zip tmp
cd tmp

#unzip all the .zip archives
find . -name "*.zip" -exec unzip '{}' \;

#delete all .zip archives
rm *.zip

#move a single shapefile (and the cooresponded .shx, .dbf, etc files)
to the "merged" directory
#(exchange 'myshape*' for the name of one of your shapefiles keeping
the '*' at the end of the name)
find . -name 'myshape*' -exec mv '{}' merged \;

#Batch merge all the remaining shapefiles from the tmp dir into the
copied file in the merge dir
#(exchange 'myshape' for the name of the copied shapefile)
for i in $(ls *.shp); do ogr2ogr -f 'ESRI Shapefile' -update -append
merged $i -nln myshape
done
```

Cet exemple de console `cmd` Windows fusionne de multiple shapefile *wetlands* dans le répertoire actuel à un seule fichier *mergedwetlands.shp* (doublez les % pour mettre dans une script , %f --> %%f) :

```
mkdir merged
for %f in (*wetland*.shp) do (
if not exist merged\wetlands.shp (
    ogr2ogr -f "esri shapefile" merged\wetlands.shp %f) else (
    ogr2ogr -f "esri shapefile" -update -append merged\wetlands.shp %f
-nln Wetlands )
)
```

Le truc est d'utiliser la première entré pour créer un nouveau shapefile puis seulement de mettre à jour. Voyez la fin de :ref:`gdal.ogr.formats.shapefile`.

Unknown interpreted text role "ref".

si vous n'avez pas besoin de définir un nom lisible par un humain via *-nln*, en utilisant *-append* est plus simple :

```
for %f in (dir1\*.shp dir2\*.shp) do (ogr2ogr -f "esri shapefile"
-append merged %f)
```

B Comment traduire un fichier de géométrie mélangée vers le format shapfile ?

Certains formats (tel que les Shapefiles d'ESRI) permet seulement un type de géométrie dans une couche, tandis que d'autres (comme les formats DGN, MapInfo, GML) permettent de mélanger des types de géométries dans une seule couche. Les tentatives de transformation résultera en des erreurs comme cela :

```
% ogr2ogr out.shp mixed.dgn
ERROR 1: Attempt to write non-linestring (POLYGON) geometry to ARC
type shapefile.
ERROR 1: Terminating translation prematurely after failed
translation of layer elements
```

La première étape dans la prise en charge de tel problème est de découvrir quels types de géométries existent dans le fichier source. Pour une fichier DGN appelé *mixed.dgn*, avec une couche appelé *elements* cela peut être accomplie en utilisant la commande SQL d'OGR (voyez [[ogr_sql]] pour les détails).

```
% ogrinfo -ro mixed.dgn -sql 'select distinct ogr_geometry from
elements'
INFO: Open of `mixed.dgn'
    using driver `DGN' successful.

Layer name: elements
Geometry: Unknown (any)
Feature Count: 1
Layer SRS WKT:
(unknown)
ogr_geometry: String (0.0)
OGRFeature(elements):0
ogr_geometry (String) = LINESTRING

OGRFeature(elements):1
ogr_geometry (String) = POLYGON

OGRFeature(elements):2
ogr_geometry (String) = POINT
```

Ce fichier a des géométries points, lignes et polygones. Chacun doit être transmis dans un fichier séparé.

```
% ogr2ogr out_point.shp mixed.dgn -where 'ogr_geometry = "POINT"'
% ogr2ogr out_line.shp mixed.dgn -where 'ogr_geometry = "LINESTRING"'
% ogr2ogr out_poly.shp mixed.dgn -where 'ogr_geometry = "POLYGON"'
```

C Comment protéger les paramètres des commandes GDAL/OGR sous la console Microsoft Windows ?

Lorsque vous travaillez avec une console Windows, il est utile de se souvenir que les règles de protection sont différent de ceux utilisé sous unix. Voici un exemple de protection correcte de requête SQL passé à ogrinfo (ou ogr2ogr) avec l'option *-sql* :

```
ogrinfo . -sql "SELECT * FROM 'B_Major Cities 1' WHERE FID = 49"
```

La requête SQL complète est entouré de guillemets doubles, mais pas de guillemets simples. Le nom de la table (ici il fonctionne pour des shapefiles ESRI dans le répertoire courant, notez le point) inclues les espaces, il doit donc être entouré avec des guillemets simples.

Par exemple, ces deux variantes présentent des commandes incorrectes :

```
ogrinfo . -sql 'SELECT * FROM 'B_Major Cities 1' WHERE FID = 49'
ogrinfo . -sql 'SELECT * FROM "B_Major Cities 1" WHERE FID = 49'
```

D Comment récupérer les formats gérés par ogr ?

Comme pour gdal, il suffit d'utiliser le paramètre --formats (avec un s ;-) :

```
ogrinfo --formats
```

renverra :

```
$ ogrinfo --formats
Supported Formats:
    -> "GRASS" (readonly)
    -> "ESRI Shapefile" (read/write)
    -> "MapInfo File" (read/write)
    -> "UK .NTF" (readonly)
    -> "SDTS" (readonly)
    -> "TIGER" (read/write)
    -> "S57" (read/write)
    -> "DGN" (read/write)
    -> "VRT" (readonly)
    -> "AVCBin" (readonly)
    -> "REC" (readonly)
    -> "Memory" (read/write)
    -> "CSV" (read/write)
    -> "GML" (read/write)
    -> "KML" (read/write)
    -> "ODBC" (read/write)
    -> "PGeo" (readonly)
    -> "OGDI" (readonly)
    -> "PostgreSQL" (read/write)
```

E Comment récupérer des informations sur la couche vectorielle ?

La commande ogrinfo permet de récupérer un grand nombre d'information d'une couche. Dans une première étape on récupère les informations contenu dans le jeu de données :

```
$ ogrinfo clc00_fr.shp
```

```
INFO: Open of `clc00_fr.shp'
        using driver `ESRI Shapefile' successful.
1: clc00_fr (Polygon)
```

Les fichiers shp possèdent toujours une couche dont le nom est le même que le nom du fichier. Certains fichiers possèdent plusieurs couches qu'il est possible de différencier par ce moyen.

Pour obtenir des informations sur une couche du jeu de données, lancez la commande suivante :

```
$ ogrinfo -so clc00_fr.shp clc00_fr
INFO: Open of `clc00_fr.shp'
        using driver `ESRI Shapefile' successful.

Layer name: clc00_fr
Geometry: Polygon
Feature Count: 270465
Extent: (37256.000205, 1607395.375873) - (1207902.000605,
2687734.936295)
Layer SRS WKT:
PROJCS["NTF_Lambert_II_étendu",
    GEOGCS["GCS_NTF",
        DATUM["Nouvelle_Triangulation_Francaise",
            SPHEROID["Clarke_1880_IGN",6378249.2,293.46602]],
        PRIMEM["Greenwich",0.0],
        UNIT["Degree",0.0174532925199433]],
    PROJECTION["Lambert_Conformal_Conic_1SP"],
    PARAMETER["False_Easting",600000.0],
    PARAMETER["False_Northing",2200000.0],
    PARAMETER["Central_Meridian",2.3372291667],
    PARAMETER["Standard_Parallel_1",45.8989188889],
    PARAMETER["Standard_Parallel_2",47.6960144444],
    PARAMETER["Scale_Factor",1.0],
    PARAMETER["Latitude_Of_Origin",46.8],
    UNIT["Meter",1.0]]
CODE_00: String (3.0)
```

Notez le paramètre `-so` qui permet de récupérer un **résumé** des informations de la couche. En absence de ce paramètre, `ogrinfo` renverra le contenu du fichier, c'est à dire les objets géographiques et la table attributaire !

F Comment écrire dans une base de données POSTGIS ?

Pour écrire dans une base de données PostGIS à partir de fichier shapefile :

```
ogr2ogr -f “PostgreSQL” PG:”host=myhost user=myloginname
dbname=mydbname password=mypassword” myshapefile.shp
```

G Comment lire une base de données au format VMAP0 ?

- Pour lire un fichier vmap0

```
ogrinfo -ro -summary gltp:/vrf/mnt/data/v0eur/vmaplv0/eurnasia
'polbnda@bnd(*)_area'
```

En cas de problème (format non reconnue), pensez à vérifier que le format OGDI est bien listé dans la liste des formats supportés. Si tel est le cas, vérifiez qu'il n'y ait pas de casse différente dans le nom du fichier (Document/Sig/ par exemple ne passera pas).

- Pour créer un fichier shp à partir d'une fichier VMAP0 :

```
ogr2ogr watrcrsl.shp gltp:/vrf/mnt/data/v0eur/vmaplv0/eurnasia
'watrcrsl@hydro(*)_line'
```

- Autre exemple, celui-ci permet de créer un fichier Shape en Lambert 2 étendue :

```
ogr2ogr -t_srs 'EPSG:27582' contourl.shp
gltp:/vrf/mnt/data/v0eur/vmaplv0/eurnasia
"contourl@elev(*)_line"
```

H Comment récupérer une zone géographique précise dans une jeu de données ?

FIXME

Le paramètre `-spat` permet de définir une bbox :

```
ogr2ogr -spat xmin ymin xmax ymax
```

Par exemple, en France :

```
ogr2ogr -spat
```

Chapitre CLII FAQ Projection et système de coordonnées

A Que sont les projections Well Known Text, et comment les utiliser ?

Well Known Texte d'OpenGIS est un format textuel pour définir les systèmes de coordonnées. Il est assez librement basé sur le modèle de système de coordonnées de l'[[http:*www.epsg.org/|EPSG]]. Bien que GDAL lui-même envoie juste ces définitions comme des chaines textuels, il y a également une classe `OGRSpatialReference <http:*www.gdal.org/ogr/classOGRSpatialReference.html>`_ dans GDAL/OGR pour les manipuler et un lien vers `PROJ.4 <http:*proj.maptools.org/>`_ pour la transformation entre les systèmes de coordonnées. La classe *OGRSpatialReference*, et le lien avec PROJ.4 (mais pas PROJ.4 lui-même) sont liés dans la bibliothèque GDAL par défaut. Plus d'informations sur WKT et *OGRSpatialReference* peuvent être trouvé dans le tutorial sur les projections dans OGR.

B Puis je reprojeter des rasters avec GDAL?

Oui, vous pouvez utiliser le programme ''gdalwarp'' ou développer un programme qui utilise la classe GDALWarpOperation décrite dans le tutorial de l'API Warp de GDAL.

C Pourquoi GDAL ne choisit pas automatiquement la transformation de datum ?

Il n'existe pas de chose comme un ensemble par défaut et précis de paramètres de transformation de datum pour un datum. OGR utilise (NADCON) par défaut, qui est le plus précis qui existe pour l'Amérique du Nord, mais dans le cas général (le monde) cela est très dur à déterminer et il n'y normalement pas de telle chose par défaut. La transformation qui doit être utilisée dépend de la zone couverte exacte, précision requise, etc. En d'autres mots, les utilisateurs doivent faire attention et faire leur boulot. Quelques liens : FAQ de proj4, Site, Message sur la liste de diffusion.

Chapitre CLIII FAQ divers

A Est ce que la bibliothèque GDAL est thread-safe ?

Non, GDAL n'est pas complètement *thread safe*.

Cependant pour GDAL 1.3.0 beaucoup de travail a été réalisé pour réaliser des scénarios thread safe. En particulier pour des situations où plusieurs threads sont lu à partir de jeux de données GDAL en même temps cela devrait fonctionner tant que deux threads n'accèdent pas le même objet *GDALDataset* en même temps. Cependant, dans ce scénario, aucun thread ne peut être écrit vers GDAL tant que les autres sont lu où cela entrainera un jolie chaos.

Aussi, bien que l'infrastructure cœur de GDAL est maintenant trhead-safe pour ce cas spécifique, seulement quelques pilotes ont été examiner pour être *thread safe*.

Il est convenue de travailler encore sur l'amélioration de la protection des thread de GDAL dans les futures versions.

B Est ce que GDAL fonctionne avec différents locales internationales pour les chiffres numérique ?

Non, GDAL se sert d'une manière intensive de *sprintf()* et de *atof()* pour traduire les valeurs numériques. Si une locale affecte le formatage des nombres, altérant le rôle de la virgule et des périodes dans les chiffres, alors PROJ.4 ne fonctionnera pas. Ce problème est commun dans certaines locales européennes.

Sur les plateformes de type-Unix, ce problème peut être évité en forçant l'utilisation de la locale des chiffres par défaut en définissant la variable d'environnement *LC_NUMERIC* à *C*, par exemple :

```
$ export LC_NUMERIC=C
$ gdalinfo abc.tif
```

C Comment débuguer GDAL ?

Différente information de débugage utile sera produit par GDAL et OGR si la variable d'environnement *CPL_DEBUG* est définie à la valeur *ON*. Lisez la documentation pour la fonction *CPLDebug()* pour plus d'information sur les messages de débugages interne.

Pour les versions plus vielles que GDAL 1.5, sur les systèmes Unix GDAL peut être compilé avec la variable d'environnement *CFG* définie pour activer la gestion du débugage avec l'option de compilation *-g*. Pour GDAL supérieur à la version 1.5, vous pouvez activer les symboles de débugages avec l'option de configuration *--enable-debug*.

Sous Windows éditez le fichier *make.opt* et assurez vous que /ZI apparaisse dans la variable OPTFLAGS.

D Comment dois je supprimer les ressources découvertes à partir de GDAL sous windows ?

La manière la plus saine pour libérer les ressources réservées et renvoyées (avec le propriétaire transféré à l'appel) à partir de la bibliothèque GDAL est d'utiliser la fonction dédiée de suppression. La suppression permet de libérer les ressources sur le bon module, sans traverser les limites des modules ce qui cause des erreurs de violation d'accès à la mémoire.

- Exemple correct de suppression de ressource :

```
OGRDataSource\* poDS = NULL;

// OGRDataSource aquisition made on side of the GDAL module
poDS = OGRSFDriverRegistrar::Open( "point.shp", FALSE );

// ...

// Properly resource release using deallocator function
OGRDataSource::DestroyDataSource( poDS );
```

- Exemple incorrect de suppression de ressource :

```
OGRDataSource\* poDS = NULL;

// OGRDataSource aquisition made on side of the GDAL module
poDS = OGRSFDriverRegistrar::Open( "point.shp", FALSE );

// ...

// Deallocation across modules boundaries.
// Here, the deallocation crosses GDAL DLL library and client's
module (ie. executable module)
delete poDS;
```

Des explications plus détaillées du problème peuvent être trouvé dans les articles suivants :

- 1ère cause possible d'erreur *segmentation fault* dans une dll
- Utiliser et libérer des la mémoire à travers des limites de modules

Chapitre CLIV Divers

Chapitre CLV Logiciel utilisant GDAL

- visualiseur de MNT 3D de MS MacroSystem
- Bluemapia : application multi-carto (Google, Microsoft, Open Street Map, NOAA/BSB Charts, raster auto-calibré) de localisation basé sur un GPS pour Windows Mobile.
- Cadcorp SIS : un SIG sous windows avec un plugin GDAL.
- CatchmentSIM : un modeleur d'analyse de terrain sous Windows pour des applications hydrologiques.
- Daylon Leveller: un modeleur de terrain/heightfield/bumpmap.
- Demeter : un autre moteur basé sur OpenGL plus ou moins similaire à VTP.
- Eonfusion : analyse et visualisation de jeux de données spatial et variant dans le temps intégrées via la "fusion de données réelles".
- ESRI ArcGIS 9.2+ : une plateforme SIG populaire.
- Feature Data Objects (FDO) : bibliothèque Open source d'accès aux données spatiales.
- flighttrack : logiciel de visualisation et de téléchargement de traces GPS pour Mac.
- FME : un paquet de traduction SIG qui inclut un plugin GDAL.
- GdalToTiles : programme C# (open source) pour réaliser des tuiles d'images pour Google Earth avec un Superoverlay KML.
- GeoDjango : un framework pour construire des applications web géographiques.
- GeoMatrix Toolkit, et GeoPlayerPro de GeoFusion : visualisation 3D.
- Duplicate explicit target name: "geoserver".
  GeoServer : un logiciel serveur open source écrit en java qui permet aux utilisateurs de partager et d'éditer des données géospatiales.
- Google Earth : un visualiseur 3D mondial.
- GRASS GIS * : un SIG raster/vecteur open source qui utilise GDAL pour l'import et l'export raster/vecteur (via r.in.gdal/ r.out.gdal et v.in.ogr/v.out.ogr).
- gstat : un paquet de modelisation géostatisque.
- gvSIG : client SIG Desktop.
- IDRISI * : une application sous Windows de traitement d'image et SIG. Utilise GDAL pour l'import/export/warp de données raster.
- ILWIS : une application Desktop de SIG et de capteurs distants.
- Image I/O-Ext : inclut gdalframework, un framework par dessus GDAL via une liaison JAVA généré par SWIG pour fournir une gestion pour un grand nombre de format de données.

- IONIC Red Spider : une plateforme de Services Web de l'OGC incluant un plugni GDAL.
- IntraMap : un logiciel SIG pour la gestion et l'affichage de données spatiales, requête spatiales et plusieurs autres analyses.
- LandXplorer : un système de visualisation 3D en temps réel pour des modèles de ville et panorama 3D.
- Leica TITAN : un environnement de visualisation et de partage de données géospatiales.
- libLAS : boîte à outil de taduction de données LiDAR ASPRS LAS 1.0/1.1 Open Source.
- libral : une implémentation d'algèbre raster et une interface graphique / console SIG expérimentale avec Perl et GTK+.
- MapGuide : serveur Web de cartographie Open source.
- Mapnik : toolkit de cartographie en C++/ Python.
- Duplicate explicit target name: "mapserver".
  MapServer : une application de cartographie web populaire avec gestion de GDAL.
- MapWindow : contrôle ActiveX open source avec des fonctionnalités SIG.
- MicroImages TNT products : logiciel avancé pour l'analyse géospatiale (Windows, LINUX, Mac OS X et UNIX)
- Mirone : paquet basé sur Matlab pour l'analyse géospatiale, océanographique et géophysique de grille.
- ogr2gui : une interface graphique utilisateur pour ogr2ogr.
- Duplicate explicit target name: "openev".
  OpenEV * : un visualiseur graphique basé sur OpenGL/GTK/Python qui utilise exclusivement GDAL pour l'accès aux raster.
- Orfeo Toolbox (OTB) : une bibliothèque générale de traitement d'image distante.
- OSSIM : un autre visualiseur et environnement d'analyse géospatiale qui utilise GDAL comme un plugin parmi plusieurs.
- Innovatin de PYXIS : une application pour la visualisation d'analyse performante et de modélisation sur des données spatiales de l'utilisateur. ''(pas encore disponible)''
- Quantum GIS (QGIS) * : un SIG bureautique multi-plateforme.
- R : un environnement logiciel libre pour le traitement statistique et graphique, en liaison avec GDAL via la paquet rgdal.
- ScanMagic : application Win32 pour la visualisation, l'analyse et le traitement de données de capteurs distants (version lite gratuite).
- Scenomics : logiciel pour compiler des bases de données de terrains utilisant GDAL pour la projection et l'import/export de données.
- SkylineGlobe : la suite Skyline des applications interactifs permet de construire, voir, lancer des requêtes et analyser des paysages virtuels 3D.
- SpaceEyes3D : logiciel de visualisation 3D pour des données cartographiques.
- Carmenta Engine (connu précédemment comme SpatialAce) : un environnement de développement d'application Rapide de SIG.
- StarSpan : analyse raster/vecteur.
- TerraGo Technologies : le format de fichier GeoPDF est utilisé pour distribuer et collaborer des données geospatiales et utilise GDAL pour l'import/export des données.
- Thuban : un visualiseur de données géographique interactif multi-platforme.
- vGeo : fusion interactive de données et outil d'analyse visuelle.
- Virtual Terrain Project * : promotion des outils pour faciliter la construction en 3D numérique du monde réel de façon interactive.

N'hésitez pas à ajouter de nouveaux logiciels dans l'ordre alphabétique, s'il vous plait écrivez une description brève. Mettez en valeur ceux qui ont une forte ou longue collaboration historique avec GDAL.

Chapitre CLVI ShapeLib

- Site : http://shapelib.maptools.org/
- Thématique : outils en ligne de commande de gestion de données au format Shapefile
- Documentation : http://shapelib.maptools.org/shapelib-tools.html

A Description

(ceci est une traduction du site).

La bibliothèque C Shapefile fournie la possibilité d'écrire de simple programme en c pour lire, écrire et mettre à jour (dans une étendue limitée) des Shapefile d'ESRI, et leur attribut associé (.dbf). Des outils à utiliser en ligne de commande sont fournies avec cette bibliothèque.

B Outils

Liste des outils :

- dbfcreate
- dbfadd
- dbfdump
- shpcreate
- shpadd
- shpdump
- shprewind
- Outils du répertoire 'contrib' de ShapeLib
  - dbfinfo
  - dbfcat
  - shpinfo
  - shpcat
  - shpcentrd
  - shpdxf
  - shpfix
  - shpproj

1 dbfcreate

- **Objectif :** créer un nouveau fichier .dbf vide.
- **Usage :** dbfcreate xbase_file [[-s field_name width],[-n field_name width decimals]]...
  - *xbase_file :* le nom du fichier xBase à créer. Pas besoin de spécifier l'extension.
  - *-s field_name width :* créer un champ *string* nommé *field_name* et une taille de *width*.
  - *-n field_name width decimals :* créer un champ numérique nommé *field_name,* de longueur *width*et avec un nombre de décimal de *decimals*.

Exemple :

```
$ dbfcreate testbase -s NAME 20, -n AREA 9 3, -n VALUE 9 2
# this will create a file named testbase.dbf with 3 fields:
NAME ( string (20)), AREA ( float (9,3)) and VALUE ( float (9,2))
```

2 dbfadd

- **Objectif :** ajoute un enregistrement dans un fichier .dbf existant.
- **Usage :** dbfadd xbase_file field_values
  - *xbase_file :* le nom du fichier xBase existant.
  - *field_values :* liste valeurs à insérer dans le fichier xBase. Vous devez spécifier un nombre de valeur égale au nombre de champ que possède le fichier xBase. L'ordre des valeurs doit également refléter l'ordre des champs dans le fichier xBase.

Exemple :

$ dbfadd testbase.dbf REGION1 25.656 150.22 # en supposant que testbase.dbf possède 3 champs (NAME, AREA and VALUE), cette commande insérera un nouvel enregistrement dans *testbase.dbf* avec les valeurs "REGION1" pour le champ NAME, '25.656' pour le champ AREA et '150.22' pour le champ VALUE.

3 dbfdump

- **Objectif :** dump le contenu d'un fichier xBase vers la console.
- **Usage :** dbfdump [-h] [-r] [-m] xbase_file
  - *-h :* affiche les informations de l'en-tête (descriptions des champs) ;
  - *-r :* résultat brute des informations des champs, valeurs numériques non reformatées ;
  - *-m :* résultat une ligne par champ ;
  - *xbase_file :* le nom d'un fichier xBase existant.

Exemple :

```
$ dbfdump -h testbase.dbf
# en supposant que testbase.dbf a un enregistrement (inséré par
l'exemple précédent en utilisant
''dbfadd''), cette commande produira les données suivantes :
```

```
Field 0: Type=String, Title=`NAME', Width=20, Decimals=0
Field 1: Type=Double, Title=`AREA', Width=9, Decimals=3
Field 2: Type=Double, Title=`VALUE', Width=9, Decimals=2
NAME AREA VALUE
REGION1 25.656 150.22
```

4 shpcreate

- **Objectif :** créé un nouveau fichier shapefile vide.
- **Usage :** shpcreate shp_file [point|arc|polygon|multipoint]
  - *shp_file :* le nom du shapefile à créer. Ne nécessite pas d'extension.
  - *point/arc/polygon/multipoint :* le type de shapefile que vous voulez créer. Vous devez définir une option valide.

Exemple :

```
$ shpcreate testpolygon polygon
# Cela créera un shapefile ponctuel nommé *testpolygon* (en fait
testpolygon.shp et testpolygon.shx
seront créé).
```

5 shpadd

- **Objectif :** ajoute un shape dans un shapefile existant.
- **Usage :** shpadd shp_file [[x y] [+]]*
  - *shp_file :* le nom d'un shapefile existant.
  - *x1 y1 x2 y2 ... xn yn :* l'ensemble des coordonnées x,y qui décrivent le shape que vous désirez ajouter. Notez que vous devez définir le nombre correcte de paramètres pour un type données de shapefile. Par exemple : pour les shapefiles ponctuels vous devez passer une paire de coordonnées XY et pour un shapfile polygonale vous devez passer au moins 4 paires de coordonnées XY (où le premier et le dernier point doivent avoir les mêmes coordonnées).

Exemple :

```
$ shpadd testpolygon 100000 7000000 250000 6500000 200000 6000000
100000 7000000
# en supposant que testpolygon est un shapefile polygonal, cette
commande insérera un nouveau shape (un
triangle) dans *testpolygon* avec les coordonnées XY suivantes :
vertice 0: 100000 7000000 (cela sera également le sommet où le shape
démarre et se termine)
vertice 1: 250000 6500000
vertice 2: 200000 6000000
vertice 3: 100000 7000000
```

6 shpdump

- **Objectif :** dump le contenu d'un shapefile en affichant l'information comme le type de shape, l'étendue du fichier, le

nombre total d'objets et les coordonnées des sommets.

- **Usage :** shpdump [-validate] shp_file
  - *-validate :* compte le nombre d'objets qui possède un ordonnancement incorrect de l'anneau.
  - *shp_file :* le nom du shapefile existant.

Exemple :

```
$ shpdump testpolygon
# en supposant que *testpolygon* est un shapefile existant déjà créé,
cette commande affichera le
résultat suivant :

    Shapefile Type: Polygon   # of Shapes: 1

    File Bounds: (  100000.000, 6000000.000,0,0)
            to  (  250000.000, 7000000.000,0,0)

    Shape:0 (Polygon)  nVertices=4, nParts=1
    Bounds:(  100000.000, 6000000.000, 0, 0)
        to (  250000.000, 7000000.000, 0, 0)
        (  100000.000, 7000000.000, 0, 0) Ring
        (  250000.000, 6500000.000, 0, 0)
        (  200000.000, 6000000.000, 0, 0)
        (  100000.000, 7000000.000, 0, 0)
```

7 shprewind

- **Objectif :** valide et reset l'ordre d'enroulement de l'anneau dans les géométries polygonales pour correspondre aux nécessités de la spécification des shapefile. Cela est utile pour les shapefile ayant des problèmes avec un 'shpdump -validate'.
- **Usage :** shprewind in_shp_file out_shp_file
  - *in_shp_file :* le nom d'un shapefile existant.
  - *out_shp_file :* le nom d'un nouveau shapefile corrigé qui sera créé.

Exemple :

```
$ shprewind badshapefile newshapefile
```

8 dbfinfo

- **Objectif :** affiche des informations basiques pour un fichier xBase donné, comme le nombre de colonne, le nombre d'enregistrement et le type de chaque colonne.
- **Usage :** dbfinfo xbase_file
- *xbase_file :* le nom d'un fichier xBase existant.

Exemple :

```
$ dbfinfo testbase

Info pour testbase.dbf
3 Columns,  1 Records in file
```

```
      NAME          string  (20,0)
      AREA           float  (9,3)
     VALUE           float  (9,2)
```

9 dbfcat

- **Objectif :** ajoute les enregistrements d'un fichier xBase source dans un fichier xBase finale. Les deux fichiers doivent avoir le même nombre de champs.
- **Usage :** dbfcat [-v] [-f] from_DBFfile to_DBFfile
- *-v :* mode verbeux.
- *-f :* force la conversion des données si les types des champs des données n'est pas le même dans les deux fichiers ou s'il y a des valeurs null dans *from_DBFfile*.
- *from_DBFfile :* fichier xBase source.
- *to_DBFfile :* fichier xBase final.

Exemple :

```
$ dbfcat -v testbase1 testbase2
```

10 shpinfo

- **Objectif :** affiche des informations basiques pour un shapefile donné, comme le type de shapefile, le nombre d'objets et leurs étendues.
- **Usage :** shpinfo shp_file
  - *shp_file :* le nom d'un shapefile existant.

Exemple :

```
$ shpinfo testpolygon

Info for testpolygon
Polygon(5), 1 Records in file
File Bounds: (         100000,        6000000)
        (         250000,        7000000)
```

11 shpcat

- **Objectif :** ajoute le contenu d'un shapfile source dans un shapefile final. Les deux fichiers doivent avoir le même type de shapefile.
- **Usage :** shpcat from_shpfile to_shpfile
- *from_shpfile :* shapefile source
- *to_shpfile :* shapefile final

Exemple :

```
$ shpcat shapefile1 shapefile2
```

12 shpcentrd

- **Objectif :** calcule le centroid XY pour des shapefile polygonaux.
- **Usage :** shpcentrd shp_file new_shp_file
  - *shp_file :* le nom d'un shapefile polygonale existant
  - *new_shp_file :* le nom d'un shapefile ponctuel qui sera créé.

Exemple :

```
$ shpcentrd apolygonfile pointcentrd
```

13 shpdxf

- **Objectif :** créé un fichier DXF à partir d'un fichier shapefile existant.
- **Usage :** shpdxf shapefile {idfield}
  - *shapefile :* le nom d'un shapefile existant.
  - *idfield : à faire*

Exemple :

```
$ shpdxf testshapefile IDFIELD
# ...
```

14 shpfix

- **Objectif :** progamme qui corrige les valeurs nulles et inconsistante dans des Shapefiles comme cela arrive de temps en temps.
- **Usage :** shpfix shpfile new_file <Record# to Blank>
  - *shpfile :* fichier en entré
  - *new_file :* fichier en sortie

Exemple :

```
$ shpfix broken fixed
```

15 shpproj

- **Objectif :** Reprojette des Shapefiles en utilisant PROJ.4
- **Usage :** shpproj shp_file new_shp (-i=in_proj_file | -i="in_params" | -i=geographic) (-o=out_info_file | -o="out_params" | -o=geographic)

15.a Entré

L'entré peut provenir d'un des trois sources. Un fichier de paramètre de projection, directement via des paramètres ou géographique. Si le shapefile possède un fichier prj, de même nom que le shapefile mais finissant par ".prj" il sera utilisé par défaut et tous les autres paramètres seront ignorés. Si l'entré est omise sa valeur par défaut est géographique, sauf si le fichier prj existe.

15.b Sortie

La sortie peut provenir d'un des trois sources. Un fichier de paramètre de projection, directement via des paramètres ou géographique. Si la sortie est omise sa valeur par défaut est géographique.

15.c Fichier de paramètres de projection

Ce fichier **doit** se terminer avec l'extension ".prj". Il est sous la forme d'un paramètre projection par ligne. Les paramètres peuvent être dans n'importe quel ordre. Les paramètres de projection sont ceux utilisé pour définir une projection PROJ.4.

15.d Paramètres de projection

Les paramètres de projection sont les mêmes que ceux utilisés par proj et invproj. Utilisez

- proj -lP : pour voir les projections disponibles
- proj -lu : pour voir les unités disponibles
- proj -le : pour voir les ellipsoïdes disponibles

Ou visitez la page web du projet PROJ.4 sur http://www.remotesensing.org/proj pour plus de détails.

Exemples :

Les exemples suivants projettent un fichier *rowtest* vers *row3,* déplace des données de *Stateplane NAD83 zone 1002* vers *utm zone 16* en mètres

```
shpproj rowtest row -i="init=nad83:1002 units=us-ft" -o="proj=utm
zone=16 units=m"

shpproj rowtest row3 -o="proj=utm zone=18 units=m" -i="zone=16
proj=utm units=us-ft"

shpproj rowtest row3 -o="proj=utm zone=18 units=m"

shpproj rowtest row3 -i=myfile.prj -o=geographic
shpproj rowtest row3 -is=myfile.prj
```

Chapitre CLVII ecwhed

Éditeur et visualiser d'en-tête ECW

A Synopsis

Usage :

```
ecwhed [--help]
   [--version]
   [-a_ullr ulx uly lrx lry]
   [-co "DATUM=VALUE"]
   [-co "PROJ=VALUE"]
   file
```

B Description

La commande `ecwhed` est un visualiseur et éditeur pour les en-têtes des fichiers ECW. Le programme peut changer les datums, la projection et les coordonnées des coins sans décompresser et recompresser l'image.

- **-a_ullr ulx uly lrx lry :** assigne/écrase les limites de géoréférencement du fichier en sortie. Cela assigne les limites de géoréférencement au fichier en sortie, ignorant celle dérivée du fichier source.
- **-co "DATUM=VALUE" :** assigne/écrase un datum
- **-co "PROJ=VALUE" :** assigne/écrase la projection

C Exemples

Par exemple, les informations de géoréférencement contenu dans un fichier ECW peuvent être affichées avec la commande suivante :

```
ecwhed FRA-0100.ecw

Datum = NTF
```

```
Projection = LM2FRANC
Origin = (223000.000000, 2152500.000000)
Pixel Size = (100.000000, -100.000000)
Corner Coordinates:
Upper Left = (223000.000000, 2152500.000000)
Upper Right = (547500.000000, 2152500.000000)
Lower Left = (223000.000000, 1690000.000000)
Lower Right = (547500.000000, 1690000.000000)
```

Par exemple, un fichier ECW peut être géoréférencé avec la commande suivante :

```
ecwhed -co "DATUM=NTF" -co "PROJ=LM2FRANC" -a_ullr 970000 1860000
980000 1850000 map.ecw
```

D Téléchargement

Le code source est disponible ici : ecwhed-0.1.tar.gz. Diffusé sous lince GPL version 2. Pour la compiler, vous avez besoin du SDK ECW d'ErMapper, qui peut être téléchargé à partir du site web d'ErMapper.

Source : http://jrepetto.free.fr/ecwhed/