Примеры использования ogr2ogr

Обсудить в форуме Комментариев — 20

Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/ogr2ogr-examples.html

Перечь примеров для справки

С библиотекой gdal поставляется утилита ogr2ogr, предназначнная для конвертации векторных данных. В данной статье приводятся примеры использования этой утилиты.

GDAL/OGR - библиотека для работы с географическими форматами данных. GDAL представляет собой набор утилит для обработки растровых данных, в то время, как OGR предназначена для работы с векторными форматами. В статье рассматриваются некоторые практические примеры применения одной из утилит этой библиотеки: программы ogr2ogr.

Содержание

Общие сведения

Программа ogr2ogr предназначена для конвертации векторных данных из одного формата в другой. Поддерживаемые форматы и используемые ключи можно узнать просто набрав в коммандной строке

```
ogr2ogr
```

В результате будет получена справка по использованию этой программы:

```
ogr2ogr --help-general -skipfailures -append -update -gt n
               -select field_list -where restricted_where
               -sql
               -spat xmin ymin xmax ymax -preserve_fid -fid FID
               -a_srs srs_def -t_srs srs_def -s_srs srs_def
               -f format_name -overwrite -dsco NAME=VALUE ...
               -segmentize max_dist
               dst_datasource_name src_datasource_name
               -lco NAME=VALUE -nln name -nlt type layer layer ...
```

А также список поддерживаемых форматов (список может отличаться как в большую, так и в меньшую сторону, поскольку зависит от того, были ли подключены/отключены соответствующие модули при компиляции программы):

- ESRI Shapefile

- MapInfo File
- TIGER
- S57
- DGN
- Memory
- BNA
- CSV
- GML
- GPX
- KML
- GeoJSON
- Interlis 1
- Interlis 2
- GMT
- SQLite
- ODBC
- PostgreSQL
- MySQL
- Geoconcept

Специфика работы с кириллицей

Чтобы ogr2ogr корректно работал с файлами в названии которых есть кириллица, необходимо, перед выполнением команд установить переменную среды:

```
SET GDAL_FILENAME_IS_UTF8=OFF
```

Если исходные данные в кодировке UTF-8, то необходимо явным образом указывать, что выходные данные также будут в UTF-8, с помощью опции `-lco ENCODING=UTF-8`, например:

```
ogr2ogr -lco ENCODING=UTF-8 output.shp input.vrt
```

Использование конфигурационных опций

Для ряда форматов через конфигурационные опции можно задать дополнительные настройки. Так, например, для формата SXF можно указать местоположение классификатора (файл RSC). Конфигурационные опции можно задавать через переменные окружения:

```
export SXF_RSC_FILENAME=map50.rsc
```

или в командной строке утилит (использование RSC из файла с другим именем):

```
ogrinfo --config SXF_RSC_FILENAME map50.rsc N-36-045.sxf
```

несколько опций задаются каждая со своим ключём (использование RSC из файла с другим именем и полные названия слоёв):

```
ogrinfo --config SXF_RSC_FILENAME map50.rsc --config SXF_LAYER_FULLNAME TRUE N-36-045.sxf
```

Конвертация

В примерах ниже будет использоваться shape-файл topo2km-rus.shp со следующими характеристиками:

```
Layer name: topo2km-rus
Geometry: Polygon
Feature Count: 10368
Extent: (-180.000000, 36.000000) - (180.000000, 88.000000)
Layer SRS WKT:
GEOGCS"GCS_Pulkovo_1942",
    DATUM"Pulkovo_1942",
        SPHEROID"Krasovsky_1940",6378245.0,298.3,
```

```
    PRIMEM"Greenwich",0.0,
    UNIT"Degree",0.0174532925199433
HEMISPHERE: String (1.0)
INDEXROMAN: String (21.0)
INDEXNUM: String (12.0)
UNIQUE: String (25.0)
ZONE: String (1.0)
I10KM: Integer (4.0)
I10KMZSTR: String (2.0)
INDEXFULL: String (50.0)
I2KM: Integer (4.0)
I2KMZSTR: String (50.0)
```

Конвертировать ESRI Shapefile в формат MapInfo TAB можно следующим образом:

```
ogr2ogr -f "MapInfo File" topo2km-rus.tab topo2km-rus.shp
```

В результате в текущем каталоге появится 4 файла topo2km-rus.* --- результат конвертации.

Теперь конвертируем только некоторые объекты из файла topo2km-rus (в поле ZONE которых содержится строка "s"):

```
ogr2ogr -f "MapInfo File" -where "ZONE=s"  topo2km-rus.tab topo2km-rus.shp
```

Аналогично можно воспользоваться параметрами:

-select для выбора определенных полей таблицы атрибутов. Например скопировать набор данных убрав все поля кроме name:

```
ogr2ogr -f GeoJSON -select name output.geojson input.geojson
```

1. -spat для выбора объектов, лежащих в определенных пространственных границах

Конвертация из CSV в shape-файл и обратно подробно рассмотрена в отдельной статье.

Работа с комплексными наборами данных

Существуют форматы позволяющие хранить несколько слоёв с разными геометриями. Например SXF. Для экстракции всех слоёв в формат ESRI Shape в качестве цели указывается папка, а не имя файла:

```
ogr2ogr output_dir input.sxf
```

Адресация конкретного слоя в таких наборах данных из нескольких слоёв адресация каждого слоя в этом случае осуществляется через пробел, например:

```
ogr2ogr output_dir input.sxf layer_name
```

Конвертация из форматов не ограниченных по типу геометрии

Для форматов, которые позволяют хранить в одном слое сразу несколько типов геометрий нужен нестандартный подход, так, чтобы конвертировать MIF/MID в формат ESRI Shape нужно указать тип геометрии, например:

```
ogr2ogr -skipfailures input_POINT.shp input.mif -nlt "POINT"
```

или, чтобы получить все типы геометрий, можно использовать цикл (Windows):

```
for %n in (POINT POLYGON) do ogr2ogr -skipfailures output_%n.shp input.mif -nlt "%n"
```

Этот подход может не подойти, так как при конвертации может происходить конвертация типов геометрии, что приведет к появлению копии данных с другой геометрией. Например, такая команда:

```
ogr2ogr -skipfailures output_LINESTRING.shp input.mif -nlt "LINESTRING"
```

Приведет к созданию линейного shape-файла, несмотря на то, что в исходном файле вообще может не быть линий, а только полигоны. Следует отметить, что преобразования между одинаковыми типами с составными объектами, наоборот, не произойдет. Т.е. MULTILINESTRING не переведется в LINESTRING, а MULTIPOLYGON в POLYGON.

В связи с этим, более правильным подходом будет явным образом указать тип данных через -sql запрос, например:

```
ogr2ogr -skipfailures output_LINESTRING.shp input.mif -nlt "LINESTRING" -sql "SELECT *
FROM Input WHERE OGR_GEOMETRY='LINESTRING'"
```

Полностью обработать данные для всех файлов с определенным расширением и всех типов геометрии можно так (цикл для командного интерпретатора Windows):

```
for %i in (*.mif) do for %n in (POINT LINESTRING POLYGON MULTIPOLYGON MULTIPOINT
MULTILINESTRING) do ogr2ogr -skipfailures %~ni_%n.shp %i -nlt "%n" -sql "SELECT * FROM
'%~ni' WHERE OGR_GEOMETRY='%n'"
```

В этом случае если в исходнике нет линий, то они не будут сконвертированы из полигонов, будет создан пустой shape-файл.

Использование OGR SQL

Можно построить и более сложные условия для выборки данных с помощью OGR SQL и параметра "-sql".

Выбрать при конвертации только объекты из таблицы "topo2km-rus", поле ZONE которых содержит 'PERVOMAYSK'

```
ogr2ogr -f "MapInfo File" -sql "SELECT * FROM topo2km-rus WHERE ZONE='PERVOMAYSK'"
topo2km-rus.tab topo2km-rus.shp
```

Выбрать отдельные поля в определённом порядке:

```
ogr2ogr -f "MapInfo File" topo2km-rus.tab topo2km-rus.shp -sql "SELECT I2KMZSTR, I2KM,
INDEXNUM FROM topo2km-rus)"
```

Выбрать и переименовать "на лету" отдельные поля в таблице

```
ogr2ogr -f "MapInfo File" topo2km-rus.tab topo2km-rus.shp -sql "SELECT I2KMZSTR AS
'i2kmzstr', I2KM AS '2km', INDEXNUM FROM topo2km-rus)"
```

Сменить тип поля "на лету" у выбранных полей (можно совмещать смену полей с их переименованием)

```
ogr2ogr -f "MapInfo File" topo2km-rus.tab topo2km-rus.shp -sql "SELECT CAST (I10KM AS
CHARACTER(4)), CAST (I2KM AS FLOAT) FROM topo2km-rus)"
```

Обратите внимание, что в SQL запросы не будут выполнены если в названиях файлов содержатся пробелы, знаки -. В этом случае название файла нужно брать в одинарные кавычки.

Значительно упростить команду можно использовав вместо ключа -sql ключ -where, его использование делает ненужным указание конструкции `SELECT * FROM 'filename' WHERE`. Пример:

```
ogr2ogr -skipfailures output_MULTIPOINT.shp input.MIF -nlt "MULTIPOINT" -where
"OGR_GEOMETRY='MULTIPOINT'"
```

вместо:

```
ogr2ogr -skipfailures output_MULTIPOINT.shp input.MIF -nlt "MULTIPOINT" -sql "SELECT *
FROM input WHERE OGR_GEOMETRY='MULTIPOINT'"
```

Обратите внимание:

- что в паре <fieldname>=<значение>, если значение - строка, то она должна браться в одинарные кавычки, как в примере выше, т.е.: -where "HIGHWAY = 'motorway'".
- после FROM требуется указать название таблицы и здесь может идти как название слоя (для ESRI Shape) так и, что-то иное для форматов с гибридными геометриями, например OGRGeoJSON (для GeoJSON).

Перепроецирование

Для перевода данных из одной системы координат в другую могут использоваться следующие параметры:

1. -a_srs, назначение системы координат набору данных, без создания нового набора (фактически, это не перепроецирование)
2. -s_srs, указание исходной системы координат (если у набора данных уже прописана система координат - она игнорируется)
3. -t_srs, требуемая система координат в которую будет осуществляться пересчет.

Например, мы знаем, что файл topo2km-rus.shp содержит данные в географической системе координат Пулково 1942, но в комплекте с topo2km-rus.shp нет файла описания проекции (*.prj). Мы можем сгенерировать этот файл, воспользовавшись командой:

```
ogr2ogr -a_srs "EPSG:4284" -f "ESRI Shapefile" topo2km-rus2.shp topo2km-rus.shp
```

Команда:

```
ogr2ogr -s_srs "EPSG:4326" -t_srs "EPSG:900913" -f "ESRI Shapefile" topo2km-rus3.shp
topo2km-rus2.shp
```

берет созданный на предыдущем этапе файл topo2km-rus2.shp и перепроецирует его в систему координат Google Mercator (epsg 900913), при этом опция -s_srs "epsg:4326" говорит о том, что при перепроецировании не нужно обращать внимания на исходную проекцию файла topo2km-rus2.shp, т.е. вести себя так, будто проекция источника - широта/долгота WGS84 (epsg 4326).

В приведенных примерах система координат указывалась на основе кодов epsg, но ее можно указывать и непосредственно в формате WKT или же передавать имя файла, в котором хранится ее описание. Например, если необходимо использовать описания систем координат, хранящихся в файлах input.prj и output.prj, то нужно использовать следующую конструкцию:

```
ogr2ogr -s_srs ESRI::Input.prj -t_srs ESRI::output.prj  shapeout.shp shapein.shp
```

Перепроецировать данные с указанием набора параметров перехода из одной системы координат в другую можно следующим образом:

```
ogr2ogr -t_srs "+proj=latlong +ellps=krass +towgs84=23.92,-141.27,-80.9,0,0,0,0"
shapeout.shp shapein.shp
```

Обрезка

Можно обрезать данные по нужному охвату, углы задаются в формате мин X, мин Y, макс X, макс Y.

```
ogr2ogr -clipsrc 41.4 46.4 41.6 46.6 output.shp input.shp
```

Вместо координат можно обрезать и по сложной форме заданной другим shape-файлом:

```
ogr2ogr -clipsrc clipbound.shp output.shp input.shp
```

или пакетно (Win):

```
for %n in (*.shp) do ogr2ogr -clipsrc clipbound.shp output_folder/%n %n
```

Объединение

Суть объединения по ogr'овски в том, что сначала создается копия первого источника данных, а потом к этой

копии добавляются данные из второго, и т.д.

```
ogr2ogr result.shp input1.shp
ogr2ogr -update -append result.shp input2.shp -nln result
```

Генерализация

Один из самых простых вариантов упрощения - отбрасывание разрядов значений координат после запятой. Ключ `-lco COORDINATE_PRECISION=1` сколько оставить разрядов после запятой.

```
ogr2ogr -f "GeoJSON" -lco COORDINATE_PRECISION=1 output.json input.shp
```

Результирующие координаты будут приведены к форме DD.D, например: 75.158028 -> 75.200000, оставшиеся нули (если нужна минификация) нужно убирать уже в текстовом представлении, например таким регулярным выражением:

```
Find: "\.0(,\)"
Replace: "$1"
```

Работа с PostreSQL/PostGIS

Вставка (добавление) записей в таблицу PostgreSQL данных из файла data.shp. Таблица должна существовать и иметь такие же поля, как и shp-файл.

```
ogr2ogr -append  -t_srs "+init=epsg:4326" -f PostgreSQL PG:"host=адрес
user=имя_пользователя dbname=имя_базы" data.shp
```

Перезаписывает test таблицу PostgreSQL данными из файла test.tab Таблица не обязана существовать.

```
ogr2ogr -append -overwrite -s_srs "+init=epsg:4326" -f PostgreSQL PG:"host=адрес
user=имя_пользователя dbname=имя_базы" test.tab
```

Перезаписывает данные из файла data в таблицу PostgreSQL. Таблица будет носить имя не data, а test1. Таблица test1 не обязана существовать.

```
ogr2ogr -append -overwrite -t_srs "+init=epsg:4326" -f "PostgreSQL"  PG:"host=адрес
user=имя_пользователя dbname=test" data.shp -nln test1
```

Наоборот: из таблицы PostgreSQL "adm" базы ipc конвертирует в allei.tab формата MapInfo.

```
ogr2ogr -f "MapInfo File" allei.tab  PG:"host=адрес user=пользователь dbname=ipc" "adm"
```

Из таблицы PostgreSQL "adm" базы ipc конвертирует в shape-файл с использованием выражения sql.

```
ogr2ogr -f "ESRi Shapefile" output.shp PG:"host=адрес user=пользователь dbname=ipc" -
sql "SELECT * from adm"
```

Обсудить в форуме Комментариев — 20

Последнее обновление: 2015-02-24 08:54

Дата создания: 21.07.2011 Автор(ы): Максим Дубинин