GDAL и кириллица в именах файлов Windows

Обсудить в форуме Комментариев — 9

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

Утилиты GDAL/OGR и кириллица в именах файлов и путей Windows. Рецепт использования кодировки UTF-8.

Содержание

Вступление:

Утилиты GDAL/OGR краткие сведения:

GDAL/OGR - это кросс-платформенная библиотека для обработки ГИС-информации, и растров, и векторов. Для программистов она представляет серьезный инструмент для создания ГИС-приложений. Богатство форматов, поддерживаемых библиотекой, позволяет использовать с огромным количеством ГИС-данных. Библиотека GDAL/OGR используется в 1 коммерческих, свободных и открытых продуктах для операций над ГИС-файлами. Не упомянутый на странице Software Using GDAL очень достойный (хотя и не очень дешевый ;)) российский продукт SCANEX IMAGE PROCESSOR®, так же использует библиотеки GDAL/OGR.

Кроме самих библиотек GDAL и OGR в инсталяционном пакете присутствует набор уже созданных утилит GDAL/OGR, запускаемых из командной строки, которые позволяют выполнять многие операции над растровыми и векторными данными. Часть утилит представлена в виде исполнимых (EXE) файлов, а часть в виде скриптов на языке Python.

О GDAL/OGR можно узнать на форуме в разделе Программное обеспечение ‹ Свободные, бесплатные, открытые ГИС ‹ GDAL/OGR. Один из вариантов установки дан в этой статье. Мне известно, что утилиты командной строки так же входят состав ГИС NextGIS QGIS. Для специалиста, умеющего общаться с командной строкой, набор утилит представляет огромные возможности, часто не сопоставимые с многими пакетами "ГИС". Сильной стороной пакета является отсутствие необходимости визуализовать файлы во время работы, что сказывается на быстродействии его операций, а так же позволяет работать с файлами, размеры которых "убивают" многие графические редакторы и ГИС-программы.

Утилиты GDAL/OGR не "любят" русские буквы в именах файлов Windows:

Это все были плюсы, к которым несомненно стоит отнести тот факт, что библиотека и утилиты открыты и бесплатны. Больше того библиотека находится в постоянном развитии. Вот с этого момента начинаются небольшие, но очень "кусачие" минусы. Останавливаться на перманентном изменении поведения отдельных частей библиотеки не имеет смысла, поскольку статья не про это. Есть люди, которые живут с этой библиотекой и ее развитием дружно, и к ним можно всегда постучаться за помощью. Проблемой является то, что часть нетривиальных случаев в поведении библиотеки приходится на пользователей Windows, которые стоят в стороне от тех, кто пишет "кросс-платформенные" библиотеки на Lunix. Из этого факта вырос неприятный сюрприз, что утилиты GDAL/OGR не "любят" русские буквы в именах путей и файлов Windows. Такая особенность возникла не сразу, а где то в процессе перехода от версии 1.6 к версии 1.9. С тех пор попытки передать программам имена с русскими буквами, а скорее всего и символами любых других национальных алфавитов, выходящих за рамки ACSII, заканчивались как то так:

```
F:\21>gdalinfo результат.tif

ERROR 4: `ЁхчєыЈЄрЄ.tif' does not exist in the file system,
and is not recognised as a supported dataset name.

gdalinfo failed - unable to open 'ЁхчєыЈЄрЄ.tif'.

F:\21>chcp 1251
F:\21>gdalinfo результат.tif

ERROR 4: `результат.tif' does not exist in the file system,
and is not recognised as a supported dataset name.

gdalinfo failed - unable to open 'результат.tif'.
```

В примере видно, что при установленной по умолчанию кодировке командной строки (CP866), вывод на экран функциями печати производится в кодировке **CP1251**, которая является кодировкой Windows, установленной в системных настройках. Этот интересный факт, пригодится нам для позже.

Решение проблемы. Костыль №1

Тема обсуждалась на нашем форуме и закончилась диагнозом. Расширенная версии обсуждения русских букв: GDAL и русские буквы в именах файлов Windows.

Вывод:

Установите переменную окружения **GDAL_FILENAME_IS_UTF8** в значение **NO**, и при работе в пределах одной кодовой страницы - будет вам счастье:

```
set GDAL_FILENAME_IS_UTF8=NO
```

Технические детали:

В процессе обсуждения такого прискорбного для русско-пишущих факта, всплыло указание на то, что эта переменная дана не от хорошей жизни, а как подпорка для тех, кто живет "неправедной" жизнью с вредными кодировками (во вредных ОС), в то время как "прогрессивное человечество" использует кодировку UTF-8, которая и встроена как основная в библиотеку GDAL/OGR. По мнению обсуждавших тему (я сам исходные коды не читал, и тут все принимаю на веру), внутренние функции GDAL/OGR оперируют строками в UTF-8, и этими же строками UTF-8 пытаются открывать файлы, если переменная окружения **GDAL_FILENAME_IS_UTF8** не установлена или имеет значение **YES**.

Так же было высказано обоснованное предположение, что Windows скорее всего имена файлов хранит в кодировке UTF-16, но потом, для пользователя, они оказывают в разных кодировках, в зависимости от места и ситуации обращения к ним. Для игрищ с кодировками консоли Windows (командный процессор **cmd**) служит команда `CHCP`. Которая как оказалось, позволяет устанавливать кодировку UTF-8 командой `CHCP 65001`. К стати, Windows 7 кодировки UTF-16 с номерами 1200 и 1201, устанавливать не позволяет.

Что можно утверждать точно, так это то, что интерпретатор языка Python 2.7, оперирует строками в кодировке UTF-8.

И все бы было хорошо, если бы не ... :
Общая неудовлетворенность запретом UTF-8:

Ну в самом деле, у всех (видимо англоязычных) есть, а нам - нельзя. А вдруг к русскому языку захочется еще немецких умляутов или другой какой UNICODE экзотики, а тут нельзя - оставайтесь только в пределах одной кодовой страницы, к тому же неизвестно где выставленной.

Проблемы со скриптами на языке Python:

Если предыдущий абзац - это блаж, без которой можно обойтись в работе, то вот такой неприятный сюрприз с вызовом утилиты '**gdal_merge.bat** из настроенной казалось системы:

```
C:\OSGeo4W64\bin>set GDAL_FILENAME_IS_UTF8
GDAL_FILENAME_IS_UTF8=NO

C:\OSGeo4W64\bin>al_merge.bat -o "F:\21\результат.tif" "F:\20 набор тестовых
файлов\x2_9140_N-37-28-Г.tif" "F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.tif"
ERROR 4: `F:\20   \x2_9140_N-37-28-.tif' does not exist in the file system,
and is not recognised as a supported dataset name.

ERROR 4: `F:\20   \x2_9140_N-37-28-.tif' does not exist in the file system,
and is not recognised as a supported dataset name.

Traceback (most recent call last):
  File "C:\OSGEO4~1\bin\gdal_merge.py", line 509, in <module>
    sys.exit(main())
  File "C:\OSGEO4~1\bin\gdal_merge.py", line 392, in main
    ulx = file_infos0.ulx
IndexError: list index out of range
```

огорчает. И "танцы с бубном": как то установка всех пришедших в голову кодовых страниц, комбинация кодовой страницы **65001** и переменной окружения **GDAL_FILENAME_IS_UTF8=YES** результата не дают:

```
@setlocal
chcp 65001
Set GDAL_FILENAME_IS_UTF8=YES
@python "%OSGEO4W_ROOT%\bin\gdal_merge.py" %*
@endlocal
```

Не работает приведенный вариант, хоть ты тресни. При этом, если вывести результат командной строки в файл, видно, что текст создается в кодировке UTF-8 (65001).

Решение проблемы. Костыль №2 (надпиленный)

Как видно выше, на самом деле BAT-файл **gdal_merge.bat** - это всего навсего обертка на python-скриптом **gdal_merge.py**. Достаточно ис имя файла, так что бы не было злосчастных русских букв, и вперед - все заработает как было задумано. Хоть с **GDAL_FILENAME_IS_UTF8=YES**, хоть **GDAL_FILENAME_IS_UTF8=NO**. Очевидное решение для этого - скопировать русский файл во временный каталог с именем без русских букв, а потом использовать эту копию.

Еще более очевидный вариант переименовать файл на время работы с ним, а потом восстановить исходное имя. Не очевидный вариант - переместить файл в пределах одного диска, так что бы ни в имени, ни в пути к каталогу, где лежит файл не было русских букв. Потому как русские буквы в имени каталога то же бывают, а переименовывать весь каталог это совсем неудобно.

Описание решаемой задачи

которая показала, что так просто с этой проблемы не обойтись. Так сложилось, что надо было мне сперва создать очень много файлов с русскими буквами, а потом еще и обработать их по всякому:

1. Были исходные данные в виде 10 больших GTIFF файлов, размером от 3 до 9 Гигабайт.
2. Файлы немного перекрывались, поскольку были результатом обработки многих сцен, собранных для трансформации.
3. Надо было разложить фрагменты этих растров по стандартным планшетам M1:50000.
4. В названиях планшетов есть русская буквы. Замена этой русской буквы на латинскую или цифру всегда приводила к путанице, да и основной пользователь продукта высказал мнение, что "русская буква есть принятый стандарт, от которого отойти - нельзя".
5. После того как все было нарезано, естественно оказалось, что отдельные планшеты, вырезанные только из одного большого растра - не полны, т.к. их фрагмент пришелся на другой растр. И эти фрагменты необходимо объединить в одни планшет.
6. Для полноты безобразия, ряд пользователей (из особо продвинутых) затребовал, что бы полученные планшеты были слиты в один растр, покрывающий некоторую область, а то ихний ...кад много мелких файлов (по 200 МБ) не понимает.

Вот для таких ритуальных танцев пришлось использовать GDAL/OGR в автономном режиме, потому как экранные ГИС умирали еще на стадии открытия одного TIF файла. И все удавалось победить, пока не дошло утилит, которые использовали скрипты на python'е. Тут работа крепко встала. Пока не было найдено решение с переименованием.

Костыль №2 (надпиленный)

Какое-то время казалось, что все хорошо. Пока не пришло время смотреть промежуточный результат. И тут стало ясно, что при сбое в утилите **gdal_merge**, переименованные файлы назад к своим именам не возвращаются. Больше того и понять как они раньше назывались дело не простое. Так что костыль оказался "надпиленный" и навернуться, опираясь на него, можно очень больно.

Вариант с копированием конечно был более безопасным, но гонять туда-сюда 60 ГБ промежуточных файлов, вышло по времени столько же, сколько работали сами программы.

Отсюда возникла задача победить таки этот зловредный UTF-8. Чему и посвящен остаток статьи.

Решение проблемы. Костыль №3

Анализ

В результате обсуждений на форуме, упомянутых выше, стало ясно, что придется смотреть коды программ, что бы понять чего там не срастается с этими русскими буквами. Началось все со скрипта **dal_merge.py**, как наиболее актуального. И быстро стало ясно, что предположение о том, что путаница начинается в момент открытия файла через функции GDAL верна только отчасти. Функция

```
gdal.Open( filename )
```

была готова открывать файлы в системной кодировке CP1251, если **GDAL_FILENAME_IS_UTF8=NO**, или в кодировке UFT-8, если **GDAL_FILENAME_IS_UTF8=YES**. Кодировку CP866, стандартную для консоли не хотела видеть ни в каком случае. Но как оказалось на вход этой функции всегда приходила строка кодированная в UTF-8, даже если по факту данные в ней не имели ничего общего с этой кодировкой.

Анализ показал, что в упомянутом скрипте, это работа вот такого кода:

```
    if argv is None:
```

```
    argv = sys.argv
argv = gdal.GeneralCmdLineProcessor( argv )
```

Не зависимо от настроек функция gdal.GeneralCmdLineProcessor считала, что разбирает строку в UTF-8. Попытки найти, что же она такое там делает, и как ее убедить так не делать, не увенчались успехом.

Но стало очевидно, что если строку с ней закомментировать, то дальше обработка полученных параметров командной строки строится на значении переменной **GDAL_FILENAME_IS_UTF8**:

1. **GDAL_FILENAME_IS_UTF8=NO** - кодовая страница должна быть CP1251;
2. **GDAL_FILENAME_IS_UTF8=YES** - кодовая страница должна быть UTF-8;

Решение получено, но ... . Опять "но"

Решение получено - надо закомментировать строку с **gdal.GeneralCmdLineProcessor**. Но тут есть пара "но":

1. ведь скрипты могут из измениться в новых версиях, и вспомнить такие подробности, именно эту строку надо "извести" - это не так просто. Хотя бы только для этого надо оставить память в виде статьи.
2. gdal.GeneralCmdLineProcessor - что-то ведь должен делать, кроме как портить входные строки. Для чего то он был задуман.

Продолжаем исследование

Пришлось искать смысл этой функции. Кода не нашел, но нашел описание идеи gdal.GeneralCmdLineProcessor и указание на то, что она входит во все утилиты (естественно все проверить мне не удалось) и задумана она для облегчения обработки сложных входных строк! И что гораздо важней, есть порождаемый этой функцией параметр

```
--optfile filename: expand an option file into the argument list.
```

который должны понимать все утилиты GDAL/ORG, и который согласно писанию предназначен для считывания параметров командной строки из файла.

Вывод

И этот параметр позволяет поместить в файл параметры в полностью контролируемой кодировке, а потом программе считать их без искажений на стыке "операционная система" - "оболочка командной строки" - "прикладаня программа"/"прикладная среда".

Решение

Оно показалось очевидным:

1. устанавливаем кодировку "приятную" нам и "оболочке командной строки". я выбрал UTF-8 или CP65001 (космополитизм, однако ...);
2. сохраняем параметры программы, получаемые из командной строки в выбранный файл в выбранной кодировке;
3. считываем это файл через параметр **--optfile filename**
4. и исправив только BAT файлы, что IMHO проще и безопасней для будущего, чем скрипты, получаем весь спектр симовлов UTF-8 в именах файлов и других праметрах.

В результате получился для **gdal_merge.py** вот такой BAT-Файл **gdal_mergeF.bat**

```
@setlocal
@echo off
@chcp 65001
Set GDAL_FILENAME_IS_UTF8=YES
set ARGV=%*
if "%1"==""  @python "%OSGEO4W_ROOT%\bin\gdal_merge.py"
```

```
if "%1"==""  goto iExit
if "%2"==""  @python "%OSGEO4W_ROOT%\bin\gdal_merge.py" %~1
if "%2"==""  goto iExit
set iTmp=%tmp%\%RANDOM%_%~n0_%RANDOM%.tmp
@rem echo %iTmp%
@echo %ARGV%>"%iTmp%"
@python "%OSGEO4W_ROOT%\bin\gdal_merge.py" --optfile "%iTmp%"
:iExit
@endlocal
```

Больше того, и скомпилированные в EXE-файлы утилиты то же через это способ можно перевести на общение через UTF-8. Для **ogrinfo.exe** получился вот такой BAT-Файл **ogrinfoF.bat**

```
@setlocal
@echo off
@chcp 65001
Set GDAL_FILENAME_IS_UTF8=YES
set ARGV=%*
if "%1"==""  @ogrinfo
if "%1"==""  goto iExit
set iTmp=%tmp%\%RANDOM%_%~n0_%RANDOM%.tmp
@rem echo %iTmp%
@echo %ARGV%>"%iTmp%"
set iTmp=%tmp%\%RANDOM%_%~n0_%RANDOM%.tmp
ogrinfo  --optfile "%iTmp%"
:iExit
@endlocal
```

Примеры:

Приведенный выше BAT файлы сработали одинаково и в случае верных данных,

- ==== EXE-файл ====

```
F:\21>ogrinfoF.bat "f:\20 набор тестовых файлов\tsp.TAB"

Active code page: 65001
Had to open data source read-only.
INFO: Open of `f:\20 набор тестовых файлов\tsp.TAB'
      using driver `MapInfo File' successful.
1: tsp (Point)
```

- ==== Py-скрипт ====

```
F:\21>gdal_mergeF.bat -o "результат.tif" "F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.tif" "F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.tif"

Active code page: 65001
0...10...20...30...40...50...60...70...80...90...100 - done.
```

и в случае, когда входные данные породили сообщение об ошибке:

- ==== EXE-файл ====

```
F:\21>ogrinfoF.bat "f:\20 набор тестовых файлов\tsp.T__"

Active code page: 65001
FAILURE:
Unable to open datasource `f:\20 набор тестовых файлов\tsp.T__' with the following
drivers.
  -> FileGDB
  -> ESRI Shapefile
  ...
```

- === Py-скрипт ===

```
F:\21>gdal_mergeF.bat -o "результат.tif" "F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.t__" "F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.t__"
```

```
Active code page: 65001
ERROR 4: `F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.t__' does not exist in the file
system,
and is not recognised as a supported dataset name.

ERROR 4: `F:\20 набор тестовых файлов\x2_9140_N-37-28-Г.t__' does not exist in the file
system,
and is not recognised as a supported dataset name.

Traceback (most recent call last):
  File "C:\OSGEO4~1\bin\gdal_merge.py", line 509, in <module>
    sys.exit(main())
  File "C:\OSGEO4~1\bin\gdal_merge.py", line 392, in main
    ulx = file_infos0.ulx
IndexError: list index out of range
```

Итог

Решение найдено. Но т.к. костыли не заменяют ног, то кроме неудобства связанно с необходимостью для всех утилит создать Bat-Файлы со специальными настройками, и помнить внесенные изменения на случай выхода новой версии утилит или библиотеки в целом, при таком подходе мы лишились возможности в ряде случаев использовать команды, начинающиеся с двойного тире "--", описанные в Gdal-dev GDAL General Command Line Processor, т.к. эти опции несовместны с опцией **--optfile filename** .

Обсудить в форуме Комментариев — 9

Последнее обновление: 2014-05-15 02:08

Дата создания: 18.12.2013
Автор(ы): Boris