

OSGeo4W Shell で はじめるコマンド処理

今日のまとめ

- Windows 版 QGIS についてくるツールだけでもいろんなコマンド処理ができるよ
- コマンド処理は覚えたり調べることも必要だけど実行する処理が決まっていれば自動化も可能だよ
- すべてがすべてコマンドで処理する必要はないけど知っている武器がひとつ増えるよ
- もっと本格的に行いたい場合 OSGeo4W Shell のほかにも方法があるよ

Windows 版 QGIS

- qgis.org から Windows 版の QGIS をインストールしようと思うと、主に二通りの案内があります

QGIS in OSGeo4W (recommended for regular users):



[OSGeo4W Network Installer](#)

Standalone installers (MSI) from OSGeo4W packages (recommended for new users)

Latest release (richest on features):



[QGIS Standalone Installer Version 3.22](#)

- そのどちらにも OSGeo4W という単語があります

OSGeo4W

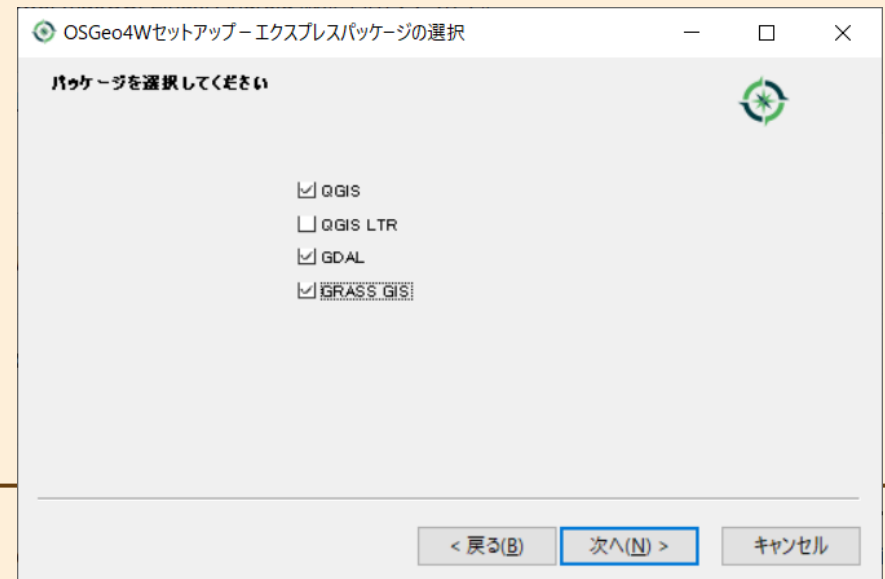
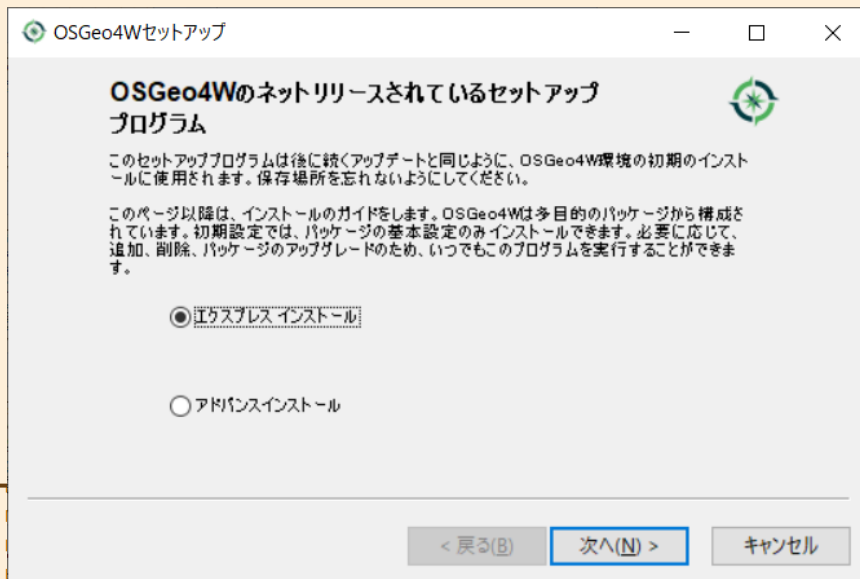
- OSGeo for Windows
- OSGeo4W は、 FOSS4G のデスクトップ GIS 、
コマンドラインで使えるユーティリティ、
ライブラリや Pythonなどを Windows で
導入しやすくまとめたパッケージ群
- 主に取り扱っているソフトウェア
 - QGIS, GRASS GIS, SAGA GIS
 - PROJ, GDAL/OGR, GEOS, SQLite3
 - Python, python-numpy, python-pandas

QGIS の二つのインストーラ

- Windows 版 QGIS はどちらも OSGeo4W のもの
- ネットワークインストーラは個別にインストール可能
 - QGIS の初期導入用インストールメニューもある
- スタンドアロンインストーラは QGIS を使うためにある程度 + α のパッケージをまとめたもの
 - OSGeo4W Shell, PROJ, GDAL/OGR, SQLite3 など同梱
 - ネットワークインストーラ (Setup) により他パッケージを追加インストールしたり、バージョンアップすることが可能

ネットワークインストーラ

- ネットワークインストーラで初めて QGIS を導入の場合
- エクスプレスインストールを選びおすすりめ設定で
 - QGIS（最新版）または QGIS LTR（安定板）
 - GDAL, GRASS GIS



スタンドアロン版→ Setup

- スタンドアロン版でインストールした場合も、
あとから任意パッケージのインストール、更新が可能
- スタートメニューの QGIS フォルダ内の Setup
- 右クリック→その他→管理者として実行
- アドバンスインストール
- インストールするフォルダ
- スタートメニューのフォルダなど指定

スタンドアロン版→ Setup

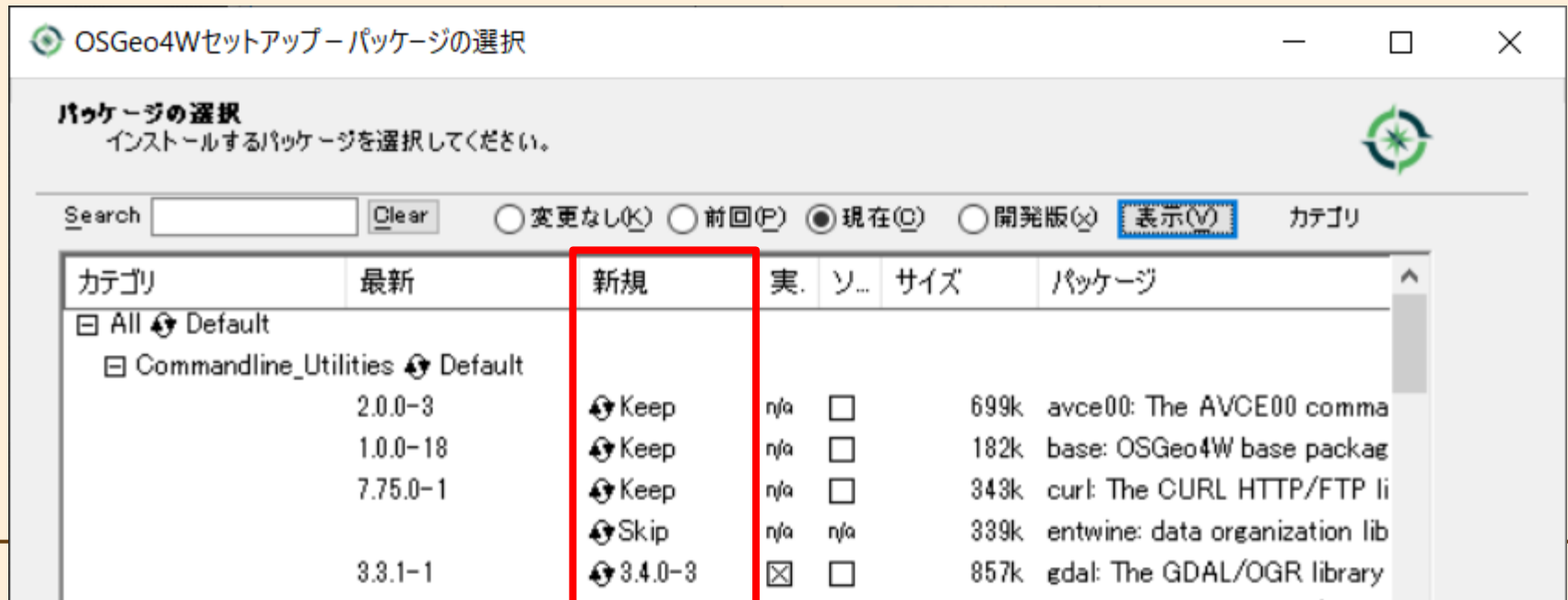
- [表示]ボタンを押すと絞り込み
 - カテゴリ すべてのパッケージをカテゴリ表示
 - 全て すべてのパッケージを名前順表示
 - 一部 更新や削除される予定のもの
 - アップデート（最新版）インストール済みで更新等不要なもの
 - 未インストール インストールされないもの



スタンドアロン版→ Setup

- 新規カラム

- Keep インストール済みで更新しない
- Skip インストールしないまま
- バージョン番号 インストール／更新する

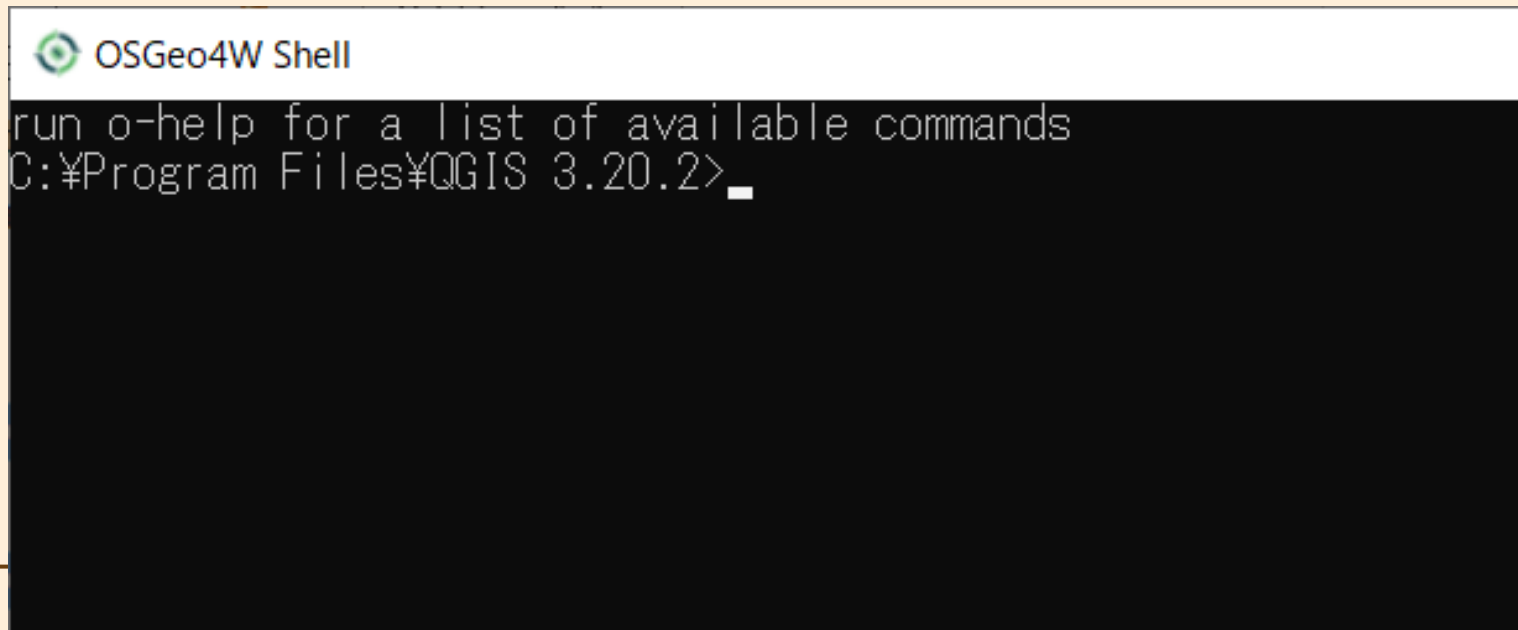


スタンドアロン版→ Setup

- 現在のバージョンを保ちつつ、新規パッケージをインストールしたい場合
 - 表示ボタン横のラジオボタンを「変更なし」
 - インストールしたい新規パッケージを探し、インストール
- ただし、トラブルが発生した場合、初期状態と環境が異なることを忘れてはいけない
- 初期インストール状態で再現するか。掲示板等で質問する場合は QGIS バージョンだけでは不適當

OSGeo4W Shell

- OSGeo4W で導入済みのツールやライブラリに対しパスを通したコマンドプロンプト
- 起動例（ QGIS 3.20.2 スタンドアロン版）

A screenshot of a Windows command prompt window titled "OSGeo4W Shell". The window has a white title bar with the OSGeo4W logo on the left. The main area is black with white text. The text shows the command "run o-help for a list of available commands" and the prompt "C:¥Program Files¥QGIS 3.20.2>_" with a cursor at the end.

```
OSGeo4W Shell
run o-help for a list of available commands
C:¥Program Files¥QGIS 3.20.2>_
```

プロンプトとコマンド

- インストールしたバージョンによって微妙に異なるが冒頭に次のような表記が表示される

```
C:\Program Files\QGIS 3.20.2>
```

- 「プロンプト」と呼ばれ、コマンド入力を促す表記
- コマンドはプロンプトのあとに入力

```
C:\Program Files\QGIS 3.20.2>echo hallo  
hallo
```

コマンド

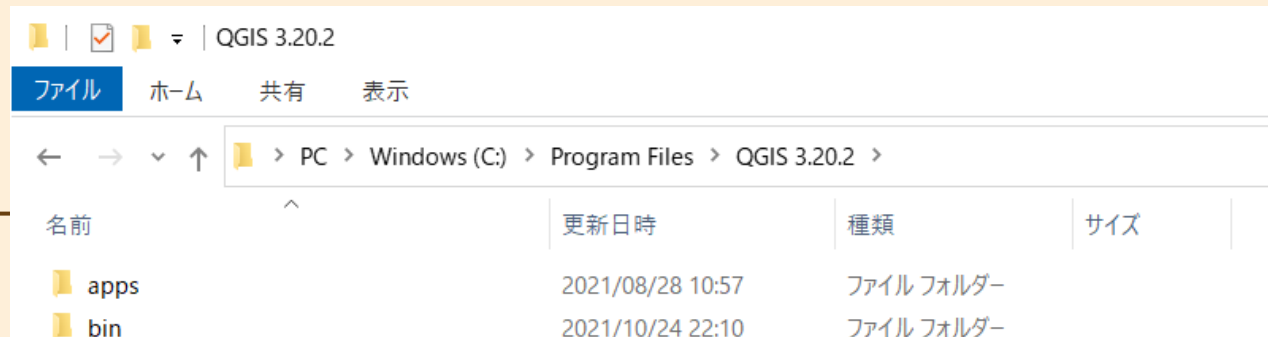
```
C:\Program Files\QGIS 3.20.2>
```

コマンドの出力結果
(出力されないものもある)

先ほどのコマンドの
処理が完了したので
次のプロンプト

プロンプトと作業フォルダ

- プロンプトの表記内容はカスタマイズでき、日時表示などに変更させることもできるが、デフォルトは現在の作業フォルダ
- 下記は C:\Program Files\QGIS 3.20.2 の中にいることを表している
 - C:\Program Files\QGIS 3.20.2>
- エクスプローラにおける次の状態



内部コマンドと外部コマンド

- コマンドプロンプトに組み込まれた内部コマンドとコマンドプロンプトから呼び出される外部コマンドがある
- 外部コマンドは実行ファイルの呼び出し
- たとえば cd コマンドは内部コマンドで ping コマンドは外部コマンド

```
C:\Program Files\QGIS 3.20.2>where cd
```

情報: 与えられたパターンのファイルが見つかりませんでした。

```
C:\Program Files\QGIS 3.20.2>where ping
C:\Windows\System32\PING.EXE
```

外部コマンドと検索パス

- コマンドプロンプトで外部実行ファイルの実行方法

- 直接呼び出す

```
C:¥Program Files¥QGIS 3.20.2>C:¥Windows¥System32¥PING.EXE
```

- 実行ファイルがあるフォルダを作業フォルダとし実行

```
C:¥Program Files¥QGIS 3.20.2>cd C:¥Windows¥System32  
C:¥Windows¥System32>PING.EXE
```

- 実行ファイルがあるフォルダを検索パスに登録

OSGeo4W Shellの検索パス

- OSGeo4W Shell は OSGeo4W でインストールした各種プログラム等を実行できるように、検索パスに追加されている

```
C:¥Program Files¥QGIS 3.20.2>set PATH  
Path=C:¥PROGRA~1¥QGIS32~1.2¥apps¥qt5¥bin;C:¥PROGRA~1¥QGIS32~1.2¥apps¥Python39¥Scripts;C:¥PROGRA~1¥QGIS32~1.2¥bin;C:¥WINDOWS¥system32;C:¥WINDOWS;C:¥WINDOWS¥system32¥WBem
```

```
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

```
C:¥Program Files¥QGIS 3.20.2>where proj  
C:¥Program Files¥QGIS 3.20.2¥bin¥proj.exe
```


コマンドの基本構文

- コマンドの基本構文は
 - コマンド [オプション] [引数]
 - オプションや引数は取らないことや省略することもある
- コマンドの種類にもよるが、対象とするものや必須となるパラメータを引数とする。
 - `mkdir demo`
 - `demo` という名前のフォルダ（ディレクトリ）を作成
 - `dir /O:D`
 - （引数が省略されているため作業フォルダ内の）
ファイル一覧を日時順に表示する

《補足》 オプションの書式

- コマンドによってオプションの書式が異なる
- コマンドプロンプトの内部コマンド
 - `cmd /O`
 - `cmd /O:P`
- Linux 系ほか多くのユーティリティ
Windows でも外部コマンドの `ping` など
 - `cmd -o`
 - `cmd -o param`
 - `cmd --opt`
 - `cmd --opt=param`

《補足》 オプションの書式

- BSD 由来のコマンド（ハイフンなし）
 - `ps aux`
 - `tar zxvf archive.tar.gz`
- GMT (General Mapping Tools)
 - `gmt coast -R-90/-70/0/20 -JM6i -B -Gchocolate`
- コマンド（プログラム）的にはオプションも引数の一種で、どう解釈し、どう処理するかはまちまち
- 慣習もあるが、結局はコマンド次第

ワイルドカード

- パターンマッチング可能な特殊文字
 - ? 任意の1文字（日本語1文字にも対応）
 - * 任意の0文字以上の文字
- 例
 - `dir *.gpkg`
 - `dir gisdata.*`

パイプライン

- コマンドの出力結果を入力として次のコマンドを実行

- 例

```
echo 135 35 | cs2cs +init=EPSG:6668 +to +init=EPSG:3857  
15028131.26      4163881.14 0.00
```

- 特に PowerShell や Linux 等では多用

リダイレクト

- コマンドの出力結果をファイルに保存
- ファイル新規作成（ > ）と追記モード（ >> ）
- 例

```
dir *.jpg >photo_list.txt
```

繰り返し（指定のセット）

- 基本構文

```
for %変数 in (セット) do (  
    処理内容  
)
```

- 特定の繰り返しセット

```
for %a in (a b c) do (  
    echo %a  
)  
a  
b  
c
```

繰り返し（数値セット）

- 周期的な数値セット

```
for /l %変数 in (start, step, end) do (  
    処理内容  
)
```

- 例

```
for /l %i in (5, -1, 2) do (  
    echo %i  
)  
5  
4  
3  
2
```


繰り返し（ファイルの各行）

- ファイルの内容に対する繰り返し

```
for /f "tokens=*" %変数 in (ファイル) do (  
    処理内容  
)
```

- 例

```
for /f "tokens=*" %l in (fruit_list.txt) do (  
    echo %l  
)  
apple  
banana  
cherry
```

繰り返し（ CSV 等のデータ）

- CSV ファイルの内容に対する繰り返し
 - for コマンドの /f オプションはデフォルトでは空白区切りで分割して処理される（行全体を得るときは tokens=* ）
 - 指定した変数を起点に順番に変数が設定される
 - %a を指定して tokens で3変数指定した場合 %a, %b, %c

```
echo EPSG:4612,EPSTG:3857 >crs.csv
for /f "tokens=1,2 delims=," %a in (crs.csv) do (
    echo 135 35 | cs2cs +init=%a +to +init=%b
)
```

コマンドの成功と失敗

- コマンドを実行すると、終了ステータスで成否がわかる
- %ERRORLEVEL% 変数
- 変数をもとに if 構文を利用し条件分岐も可能
- `command1 && command2`
 - `command1` が成功したら `command2` を実行
 - `command1` が失敗したら `command2` は実行しない
- `command1 || command2`
 - `command1` が成功したら `command2` は実行しない
 - `command1` が失敗したら `command2` を実行

《余談》 and と or

- && は論理積（かつ； and ）
- 両方「成功」でないと式全体として「成功」といえない
- `command1 && command2`
 - `command1` が成功しても `command2` を実行して成功するか確認しないと式全体の評価が下せない
 - `command1` が失敗したら、式全体として既に「失敗」であり、`command2` は実行する必要がない
- 例（ Linux ）
 - `./configure && make && make install`

《余談》 and と or

- `||` は論理和（または ; or）
- 片方でも「成功」なら式全体として「成功」
- `command1 || command2`
 - `command1` が成功したら、式全体として既に「成功」であり、`command2` は実行する必要がない
 - `command1` が失敗したら、`command2` を実行して成功するか確認しないと式全体の評価が下せない
 - いわゆる（？） Trick or Treat 構文
- 例（Perl 言語）
 - `open(FILE, "file.txt") or die("open error");`

バッチファイル

- 何度も実行したい規定の処理
- 複数行にまたがる複雑で覚えるのが大変な処理など
- コマンドを記述したバッチファイルとして保存し実行
- エクスプローラからダブルクリックで実行することもできるが、OSGeo4W のパスが通っていない、ただのコマンドプロンプトのバッチと認識される
- バッチファイル内でパスを通す
- もしくは OSGeo4W Shell からバッチファイルを呼び出す
 - call ファイル名.bat

バッチファイルの注意点

- % 記号は構文解析のタイミングで特殊な扱い
 - 単独 % は認識されない
 - for 構文などでは %i と記述すると %i と認識される
 - つまりバッチファイルで for %f in (*.txt) と書くとエラー
- コマンドプロンプトで for 構文について検索すると %i という記述のサイトと %%i という記述のサイトがあるので注意が必要
- 各コマンドでプロンプトが表示されるのが嫌なとき
 - @echo off

OSGeo4W Shell のコマンド

- OSGeo4W でインストールしたパッケージにより使用できるコマンドは異なる。追加も可能
- 利用可能なコマンドは o-help コマンドで確認できる
 - run o-help for a list of available commands
- ただし o-help では %OSGEO4W_ROOT%\bin の中にある実行ファイルやバッチファイルのみで、パスが通った他のファイルは一覧に出てこない
- たとえば gdal2xyz.bat （実態は gdal2xyz.py ）
 - dir %OSGEO4W_ROOT%\apps\Python39\Scripts*.bat

PROJ

- 座標値の変換計算ユーティリティ/ライブラリ
- QGISをはじめ多くの GIS ツールで利用されている
- コマンドラインユーティリティはいくつかあるが
 - projinfo
 - 座標系などの情報を表示する
 - cs2cs
 - 座標値の座標系間の変換計算を行う
 - <https://proj.org/apps/index.html>

projinfo

- 座標系などの情報を表示する
- 例（出力結果略）
 - `projinfo EPSG:3857`
 - EPSG:3857 の情報を問い合わせ
 - `projinfo -s EPSG:4612 -t EPSG:6677`
 - EPSG:4612 から EPSG:6677 への変換の手法について問い合わせ

cs2cs

- 座標値の座標系間の変換計算を行う
- 基本構文
 - `cs2cs` [オプション] 変換元SRS +to 変換先SRS [ファイル]
 - 座標ファイルを指定しないときは標準入力から座標値を入力
 - 標準入力待ちの状態を終了したいときは Ctrl+C
 - `echo 座標 | cs2cs` [オプション] 変換元SRS +to 変換先SRS
 - `echo` コマンドで座標値を出力し、パイプラインでつなぐことで、標準入力待ちを行わないこともできる
- SRS (CRS) はいわゆる Proj 記法を用いる

cs2cs

- 座標値の座標系間の変換計算を行う
- 例（出力結果略）
 - `echo 135 35 | cs2cs +init=EPSG:6668 +to +init=EPSG:6673`
 - EPSG:6668 (JGD2011 longlat) から EPSG:6673 (JGD2011 平面直角座標系 V 系) へ変換
 - `echo 135 35 >points.txt`
`echo 140 40 >>points.txt`
`cs2cs +proj=longlat +ellips=GRS80 +to`
`+proj=geocent +ellips=GRS80 points.txt`
 - points.txt 内の座標を、経緯度から地心直交座標系へ

GDAL/ORG

- GIS データの形式変換、座標系変換など
- また派生プログラムによる GIS 処理も多数
- ラスタ系が GDAL 、ベクタ系が OGR
- コマンドラインユーティリティはいくつかあるが
 - gdalinfo
 - ラスタデータの情報を表示する
 - gdal_translate
 - 異なる形式への変換や解像度の変換などを行う
 - gdalwarp
 - 異なる座標系への投影変換を行う

GDAL/ORG

- ラスタ系が GDAL 、ベクタ系が OGR
 - gdal_grid
 - （散布状の）点データからラスタデータを作成する
 - gdal2xyz
 - ラスタデータから ASCII xyz 形式に変換する
 - ogrinfo
 - ベクタデータの情報を表示する
 - ogr2ogr
 - 異なる形式への変換や異なる座標系への投影変換を行う

SQLite3

- 単ファイルのデータベース
- GeoPackage / Spatialite 形式や、PROJ の proj.db 、 QGIS のプロファイル設定 qgis.db などでの利用
- 知識があれば、トラブル時に直接データの中身を確認することもできる
- SQL を利用した GIS 処理なども可能（集計等に強い）
 - GIS 処理を行う場合は mod_spatialite を読み込む必要がある

SQLite3 の実行

- 基本構文
 - `sqlite3 [オプション] [データベースファイル] [SQL]`
 - SQL を指定しない場合は対話型で起動
 - SQLite 用のプロンプト `sqlite>` が表示される
 - `.quit` で終了

- 例

```
C:¥>sqlite3 test.sqlite
sqlite> SELECT load_extension("mod_spatialite");
sqlite>
sqlite> .quit
```


Python

- 近年大人気のスクリプト言語
- QGIS プラグイン等で使われるほか、GDAL/OGR など Python API に対応しているライブラリも多い
 - 機能の本体は C++ 等で書かれているが、Python から機能呼び出すことで、簡易に利用することができる
- Python 言語で独自に処理を記述し実行することも、他者が作成した Python ツールを実行することもできる
- ライブラリ追加は OSGeo4W Setup で追加できるものはそちらで導入する方が楽。それ以外も pip 等で可能
 - fiona, geopandas など

Python の実行

- QGIS 3.18 版以前の同梱 OSGeo4W Shell では Python 関係のパスが通っていない
 - パスを通すため py3_env を実行する必要がある
 - 3.20 以降は逆に py3_env は（見えるところに）存在しない
- 基本構文
 - python3 [オプション] [スクリプトファイル] [引数]
 - スクリプトファイルを指定しない場合は対話型で起動
 - Python 用のプロンプト >>> が表示される
 - quit() で終了

ハンズオン

コマンドプロンプトのコピー

- ハンズオンを行う前にコピーの仕方について
- コマンドプロンプト内のテキストをコピーするには
 - ドラッグして範囲選択 → 右クリック
- クリップボードをコマンドプロンプトに貼り付けるには
 - コマンドプロンプトが選択モードの場合 ESC キーで解除
 - 右クリック
- 選択モードになっている場合、コピーと扱われるためペーストできない&クリップボード上書きされるので注意

データのダウンロード

- 使用するデータやコピペ可能なコマンドファイルをダウンロードしてください
 - https://github.com/tohka/FOSS4G2021_Tutorial
 - 右上の Code ボタンを押し Download ZIP
 - ダウンロードしたら展開してください（場所は任意）
- ダウンロードファイル内に含まれている GIS データはパブリックドメイン下で公開されている Natural Earth、もしくはそれを加工したデータです
 - <https://www.naturalearthdata.com/>

OSGeo4W Shell の起動

- OSGeo4W Shell を起動してください
 - QGIS のバージョンなどは問いません
 - ただし QGIS 3.18 版以下は py3_env を実行してください
- 先ほどダウンロードし、展開したフォルダのアドレスをコピーし OSGeo4W Shell で下記コマンドを実行してください
 - `cd "フォルダのアドレス"`
- プロンプトが変化し、作業フォルダが目的のフォルダになったことを確認してください

1. ベクタデータの確認

- ベクタデータの情報を確認します

```
cd tutorial_01  
dir
```

```
ogrinfo ne_50m_land.shp  
ogrinfo ne_50m_land.gpkg  
ogrinfo ne_50m_land.json
```

- 基本構文
 - ogrinfo [オプション] データソース [レイヤ]

1. ベクタデータの確認

- ベクタデータの情報を確認します

```
ogrinfo -so ne_50m_land.shp ne_50m_land
```

```
ogrinfo -so ne_50m_land.gpkg ne_50m_land
```

```
ogrinfo -so ne_50m_land.json ne_50m_land
```

- -so
 - summary only
 - つけないと各地物の詳細も表示される

2. ベクタデータの形式変換

- ベクタデータの形式を変換します

```
cd tutorial_02  
dir
```

```
ogr2ogr -f GPKG ne_50m_land.gpkg ne_50m_land.shp
```

- 基本構文

- ogr2ogr [オプション] 出力データ 入力データ [レイヤ]
- OGR コマンド共通オプション、ogr2ogr 固有オプション、入力形式固有オプション、出力形式固有オプションがある
- https://gdal.org/programs/vector_common_options.html
- <https://gdal.org/programs/ogr2ogr.html>
- <https://gdal.org/drivers/vector/index.html>

2. ベクタデータの形式変換

- 形式固有の warning がでたりすることもある
- GeoPackage の場合、ポリゴン形式でマルチポリゴンを保持することはダメ（ESRI Shapefile はポリゴン形式）
- 追加で入力データを開く際の文字列エンコーディングを明示するオプションを追加
（デフォルトが UTF-8 なので今回はなくてもよい）

```
ogr2ogr -f GPKG -nlt PROMOTE_TO_MULTI -oo ENCODING=UTF-8  
ne_50m_land.gpkg ne_50m_land.shp
```

3. ベクタデータの再投影

- ベクタデータの再投影をします

```
cd tutorial_03  
dir
```

```
ogr2ogr -f GPKG -s_srs EPSG:4326 -t_srs EPSG:3857  
ne_50m_land_3857.gpkg ne_50m_land.gpkg
```

- 投影法の指定
 - s_srs 入力データの SRS (CRS) の指定
データ自身に座標系情報を保持している場合は不要
 - t_srs 変換先の SRS (CRS)

4. for を使った一括処理

- for 文を使って一括処理をします

```
cd tutorial_04  
dir
```

```
for %f in (*.shp) do (  
    ogr2ogr -f FlatGeobuf -nlt PROMOTE_TO_MULTI %~nf.fgb %f  
)
```

- 変数の切り出し
 - 変数が %f のとき %~nf で拡張子を切り取った文字列
 - このほかいろいろある

5. ラスタデータの確認

- ラスタデータの情報を確認します

```
cd tutorial_05  
dir
```

```
gdalinfo NE1_50M_SR_W.tif  
gdalinfo NE1_50M_SR_W.jpg  
gdalinfo NE1_50M_SR_W_no_aux.jpg  
gdalinfo NE1_50M_SR_W_no_wld.jpg
```

- オリジナルデータ（のリサイズ版）が TIFF ファイル
- そのファイルから JPEG に変換し、各種ファイルを取り除いたものになります

5. ラスタデータの確認

- TIFF 形式はタグ情報を埋め込むことができる画像形式
 - 位置情報を含む TIFF を特に GeoTIFF という
 - また色情報を扱う一般の画像と異なり、整数値ではなく実数値をとることもできる（標高データなど）
-
- 逆に JPEG 形式などは位置情報を扱えない
 - ESRI Worldfile 形式などの別ファイルで位置をもつ
 - ただ Worldfile は位置を座標値で位置を示すが、それがどの座標系の座標値かは保持できない

6. ラスタデータの形式変換

- ラスタデータの情報を確認します

```
cd tutorial_06  
dir
```

```
gdal_translate -of JPEG -co WORLDFILE=YES NE1_50M_SR_W.tif  
NE1_50M_SR_W.jpg  
dir
```

- 基本構文

- `gdal_translate -of 出力形式 [オプション] 入力データ 出力データ`
- GDAL コマンド共通オプション、`gdal_translate` 固有オプション、入力形式固有オプション、出力形式固有オプションがある

6. ラスタデータの形式変換

- 共通オプション
 - https://gdal.org/programs/raster_common_options.html
- gdal_translate
 - https://gdal.org/programs/gdal_translate.html
- GeoTIFF ドライバ
 - <https://gdal.org/drivers/raster/gtiff.html>
- JPEG ドライバ
 - <https://gdal.org/drivers/raster/jpeg.html>
- -co WORLDFILE=YES が JPEG 固有の作成オプション

7. ラスタデータの位置指定

- 位置情報を持っていないラスタデータに位置を与えます

```
cd tutorial_07  
dir
```

```
gdalinfo NE1_50M_SR_W_no_wld.jpg
```

```
gdal_translate -of GTiff -a_srs EPSG:4326 -a_ullr -180 90  
180 -90 NE1_50M_SR_W_no_wld.jpg NE1_50M_SR_W_with_pos.tif
```

```
gdalinfo NE1_50M_SR_W_with_pos.tif
```

7. ラスタデータの位置指定

- 指定すべき位置情報は既知
 - 指定した SRS (CRS) を割り当てる
 - -a_srs オプション (assign SRS)
 - 四隅の座標を指定する
 - -a_ullr オプション (assing upper-left, lower-right)
- gdal_translate はこのほかデータ型の変換、大きさ（解像度）の変更およびリサンプリング、各ファイル形式固有の設定（圧縮方法など）の変更
- 投影変換はできない

8. ラスタデータの投影変換

- ラスタデータを投影変換します

```
cd tutorial_08  
dir
```

```
gdalwarp -of GTiff -t_srs EPSG:3832 -te -200000000 -200000000  
200000000 200000000 -ts 2000 2000 -r cubic NE1_50M_SR_W.tif  
NE1_50M_SR_W_3832.tif
```

- オプション

- -te 範囲（地図座標；デフォルトは投影変換後の座標）
- -ts 出力画像の縦横サイズ
- -r リサンプリング手法

9. Python スクリプトの実行

- Python スクリプトを利用してファイルダウンロードし、複数のタイル画像を扱います

```
cd tutorial_09  
dir
```

```
python3 download_tiles.py  
dir
```

```
gdalbuildvrt -a_srs EPSG:3857 tiles.vrt *.png
```

```
gdalwarp -t_srs "+proj=lcc +ellps=GRS80 +lon_0=137  
+lat_1=34.083333 +lat_2=37.916667" tiles.vrt japan_lcc.tif
```

10. 外部ツールの利用

- Python で書かれた OSS のツールを利用します
- 下記ページよりダウンロード、ZIP 展開後、tutorial_10 下に移動させてください
 - <https://github.com/mapbox/mbutil>

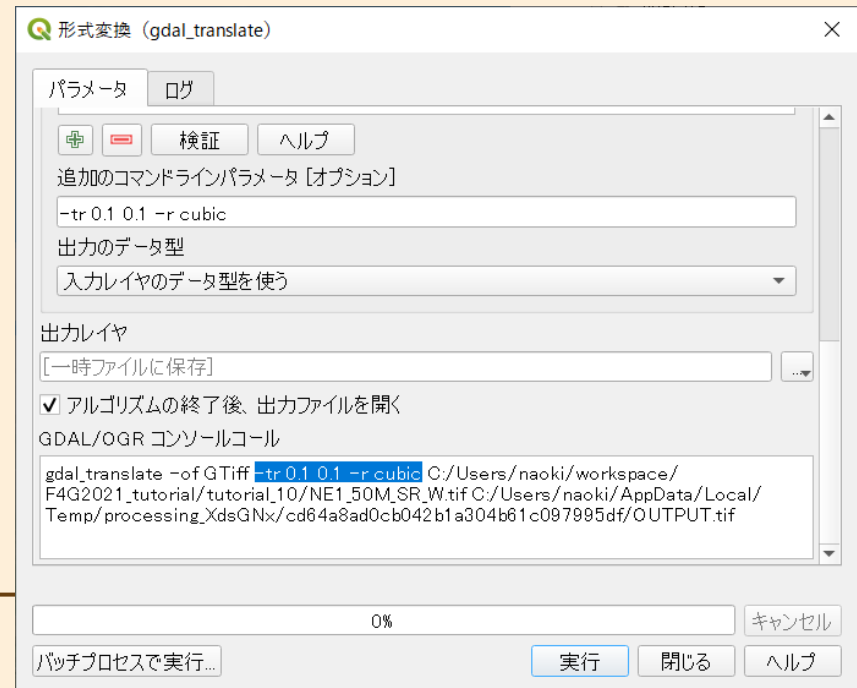
```
cd tutorial_10  
dir
```

```
gdal2tiles --zoom=1-4 --xyz NE1_50M_SR_W.tif tiles  
python3 generate_metadata.py tiles
```

```
python3 mbutil-master¥mb-util tiles tiles.mbtiles
```

QGIS での活用について

- QGIS では GDAL/OGR などを活用しています
- 内部処理だけでなく、プロセッシングツールの各処理で直接呼び出しているため、ダイアログ内で指定できない項目をコマンドラインパラメータとして与えることもできます。



OSGeoLive / WSL

- OSGeo4W Shell は Windows で使えるツールがある
- Python とか R とかも本格的に行うなら環境構築可
- でも、もっと本格的にコンソール処理をやるなら...
 - Mac / Linux
- 特に Linux はフリーで構築可能
 - VirtualBox などの環境でホスト
 - 最近では WSL (Windows Subsystem for Linux)
 - かなりシームレスに Windows 上で実行できる

今日のまとめ

- Windows 版 QGIS についてくるツールだけでもいろんなコマンド処理ができるよ
- コマンド処理は覚えたり調べることも必要だけど実行する処理が決まっていれば自動化も可能だよ
- すべてがすべてコマンドで処理する必要はないけど知っている武器がひとつ増えるよ
- もっと本格的に行いたい場合 OSGeo4W Shell のほかにも方法があるよ