State of GDAL
GDAL 3.10 & 3.11

Even Rouault

SPATIALYS

2024 GDAL user survey

- 600 respondents representing industry, government, academia
- Invaluable information about
  - Where do people obtain GDAL?
  - What platforms?
  - What tools do they use?
  - What would make GDAL better?
- 2 main findings:
  - Need for more documentation (understanding features, finding examples, API documentation) ⇒ 2 part-time documentation writers funded by GSP
  - Command line usage of GDAL by more than 50% of respondants

Status of current GDAL utilities

- Powerful and comprehensive
- 20+ years of backwards compatibility
- But many inconsistencies:
  - Underscore or not? `gdal_translate` vs `gdalwarp`
  - Dataset order: `gdal_translate in.tif out.tif` vs `ogr2ogr out.gpkg in.gpkg`
  - Argument naming: `gdalwarp -ts width height` vs `gdal_translate -outsize width height`
  - Dash vs double dash: `gdal_calc --type` vs `gdal_translate -ot`
- Lots (too many) options ⇒ ogr2ogr: 76 switches!

RFC 104: Adding a `gdal` CLI

- Burn the village to the ground 🔥
- `git` -style sub-commands
- Break apart huge CLIs
  - `gdal_translate` and `ogr2ogr`
  - `gdalwarp` and `gdaldem`
- Unified CLI framework
- Programmatic discoverability
- GDALG pipelining

`gdal`: what's new? (1 / 2)

- Short options ⇒ single dash: `-i my.tif`
- Long options ⇒ double dash: `--update`
- Two possibilities to pass values:
  `--input=my.tif` or `--input my.tif`
- Input(s) first, output last:
  ```
  $ gdal raster convert in.gpkg out.tif
  ```
- Always use hyphen (no more _'s)

`gdal`: what's new? (2 / 2)

- Never overwrite output without `--overwrite`
- Detection of typos such as letter inversion

```
$ gdal raster convret
ERROR 1: Algorithm 'convret' is unknown. Do you
mean 'convert'?
```

```
$ gdal raster convert --creattionoption
ERROR 5: convert: Option '--creattionoption' is
unknown. Do you mean '--creation-option'?
```

First level commands

```
$ gdal

ERROR 1: gdal: Missing command name.
Usage: gdal <COMMAND> [OPTIONS]
where <COMMAND> is one of:
  - convert:  Convert a dataset (shortcut for 'gdal raster convert' or 'gdal
vector convert').
  - driver:   Command for driver specific operations.
  - info:     Return information on a dataset (shortcut for 'gdal raster info'
or 'gdal vector info').
  - mdim:     Multidimensional commands.
  - pipeline: Execute a pipeline (shortcut for 'gdal raster pipeline' or 'gdal
vector pipeline').
  - raster:   Raster commands.
  - vector:   Vector commands.
  - vsi:      GDAL Virtual System Interface (VSI) commands.

Try 'gdal --help' for help.

'gdal <FILENAME>' can also be used as a shortcut for 'gdal info <FILENAME>'.
And 'gdal read <FILENAME> ! ...' as a shortcut for 'gdal pipeline <FILENAME> !
...'.

For more details, consult https://gdal.org/programs/index.html
```

gdal raster (1 / 2)

```
- aspect:        Generate an aspect map  (gdaldem aspect)
- calc:          Perform raster algebra
- clean-collar:  Clean the collar, removing noise. (nearblack)
- clip:          Clip a raster dataset   (gdal_translate -projwin)
- color-map:     Generate a RGB or RGBA dataset from a single band, using
                 a color map (gdaldem color-relief or gdal_translate -expand rgb)
- contour:       Creates a vector contour from a raster elevation model (DEM)
- convert:       Convert a raster dataset (gdal_translate)
- create:        Create a new raster dataset (gdal_create)
- edit:          Edit a raster dataset (gdal_translate -mo/-a_nodata/-a_srs)
- fill-nodata:   Fill nodata raster regions by interpolation from edges.
                 (gdal_fillnodata.py)
- footprint:     Compute the footprint of a raster dataset (gdal_footprint)
- hillshade:     Generate a shaded relief map (gdaldem hillshade)
- index:         Create a vector index of raster datasets (gdaltindex)
- info:          Return information on a raster dataset (gdalinfo)
- mosaic:        Build a mosaic, virtual (VRT) or materialized. (gdalbuildvrt)
- overview:      Manage overviews of a raster dataset (gdaladdo)
- pipeline:      Process a raster dataset.
- pixel-info:    Return information on a pixel (gdallocationinfo)
- polygonize:    Create a polygon feature dataset from a band (gdal_polygonize)
- reclassify:    Reclassify values in a raster dataset
- reproject:     Reproject a raster dataset (gdalwarp)
```

gdal raster (2 / 2)

- resize: Resize a raster dataset without changing the georeferenced extents *(gdal_translate -outsize)*
- roughness: Generate a roughness map *(gdaldem roughness)*
- scale: Scale the values of the bands *(gdal_translate -scale)*
- select: Select a subset of bands *(gdal_translate -b)*
- set-type: Modify the data type of bands *(gdal_translate -ot)*
- sieve: Remove small polygons *(gdal_sieve.py)*
- slope: Generate a slope map *(gdaldem slope)*
- stack: Combine together input bands into a multi-band output, virtual (VRT) or materialized *(gdalbuildvrt -separate)*
- **tile**: Generate tiles in separate files from a raster dataset. **(enhanced gdal2tiles)**
- tpi: Generate a Topographic Position Index (TPI) map *(gdaldem TPI)*
- tri: Generate a Terrain Ruggedness Index (TRI) map *(gdaldem TRI)*
- unscale: Convert scaled values of a raster dataset into unscaled Values. *(gdal_translate -unscale)*
- viewshed: Compute the viewshed of a raster dataset *(gdal_viewshed)*

`gdal raster tile`

- C++ port of `gdal2tiles.py`
- Enhancements
  - Handle non-Byte data types (UInt16, Float32…)
  - More tiling schemes
    - consecutive overview levels with resolution != 2 (ie NZTM2000)
    - alternative origin
    - different tile size
  - 3x to 6x faster
  - Unified progress report

gdal vector

- clip: Clip a vector dataset *(ogr2ogr -clipsrc / -clipdst)*
- concat: Concatenate vector datasets *(ogr_merge.py)*
- convert: Convert a vector dataset *(ogr2ogr)*
- edit: Edit metadata of a vector dataset *(ogr2ogr -a_srs/-mo)*
- filter: Filter a vector dataset *(ogr2ogr -spat/-where)*
- geom: Geometry operations on a vector dataset *(ogr2ogr …)*
- grid: Create a regular grid from scattered points *(gdal_grid)*
- info: Return information on a vector dataset *(ogrinfo)*
- pipeline: Process a vector dataset
- rasterize: Burns vector geometries into a raster *(gdal_rasterize)*
- reproject: Reproject a vector dataset *(ogr2ogr -t_srs)*
- select: Select a subset of fields *(ogr2ogr -select)*
- sql: Apply SQL statement(s) to a dataset *(ogr2ogr -sql)*

GDALG: generating pipelines

- **Some raster or vector algorithms accept streamed input and generate streamed outputs ⇒ processing pipelines**

```
$ gdal raster pipeline read in.tif ! \
    reproject --dst-crs=EPSG:4326 ! \
    edit --metadata TITLE=my-title ! \
    write out.tif --of COG
```

```
$ gdal vector pipeline ! \
    read in.gpkg ! \
    select fields=name,geometry ! \
    reproject --dst-crs=EPSG:4326 ! \
    geom make-valid ! \
    clip --box=2,49,3,50 ! \
    write out.gpkg --overwrite
```

GDALG: serializing pipelines

- **Generalization of GDAL or OGR VRT**
- **GDALG format just records the command line**

```
$ gdal raster pipeline read in.tif ! \
   reproject --dst-crs=EPSG:4326 ! \
   edit --metadata TITLE=my-title ! \
   write out.gdalg.json

$ cat out.gdalg.json
{
  "type":"gdal_streamed_alg",
  "command line":"gdal raster pipeline read --input
in.tif ! reproject --dst-crs EPSG:4326 ! edit
--metadata TITLE=my-title",
  "gdal_version": "3110000"
}
```

GDALG: replaying pipelines

```
$ gdal info out.gdalg.json --of=text
Driver: GDALG/GDAL Streamed Algorithm driver
Files: out.gdalg.json
Size is 16384, 8192
Coordinate System is:
GEOGCRS["WGS 84",
[...]
    ID["EPSG",4326]]
Origin = (-180.000000000000000,90.000000000000000)
Pixel Size = (0.021972656250000,-0.021972656250000)
Metadata:
  TITLE=my-title
Corner Coordinates:
[..]
Band 1 Block=512x128 Type=Byte, ColorInterp=Red
Band 2 Block=512x128 Type=Byte, ColorInterp=Green
Band 3 Block=512x128 Type=Byte, ColorInterp=Blue
```

Bash completion

- **Command and subcommand completion**

```
$ gdal <TAB><TAB>
convert   driver  info  mdim  pipeline  raster
   vectorvsi
```

- **Option name completion**

```
$ gdal raster convert --<TAB><TAB>
--append          --creation-option  --if           --input-format
--oo              --output           --overwrite    --co
--format          --input            --of           --open-option
--output-format   --progress
```

Bash completion

- **Option value completion**

```
$ gdal raster convert --of=<TAB><TAB>
AAIGrid         DDS             FITS            GTiff
AVIF            DTED            GeoRaster       GTX
BAG             ECW             GIF             HDF4Image
BASISU          EHdr            GPKG            HEIF
[...]
```

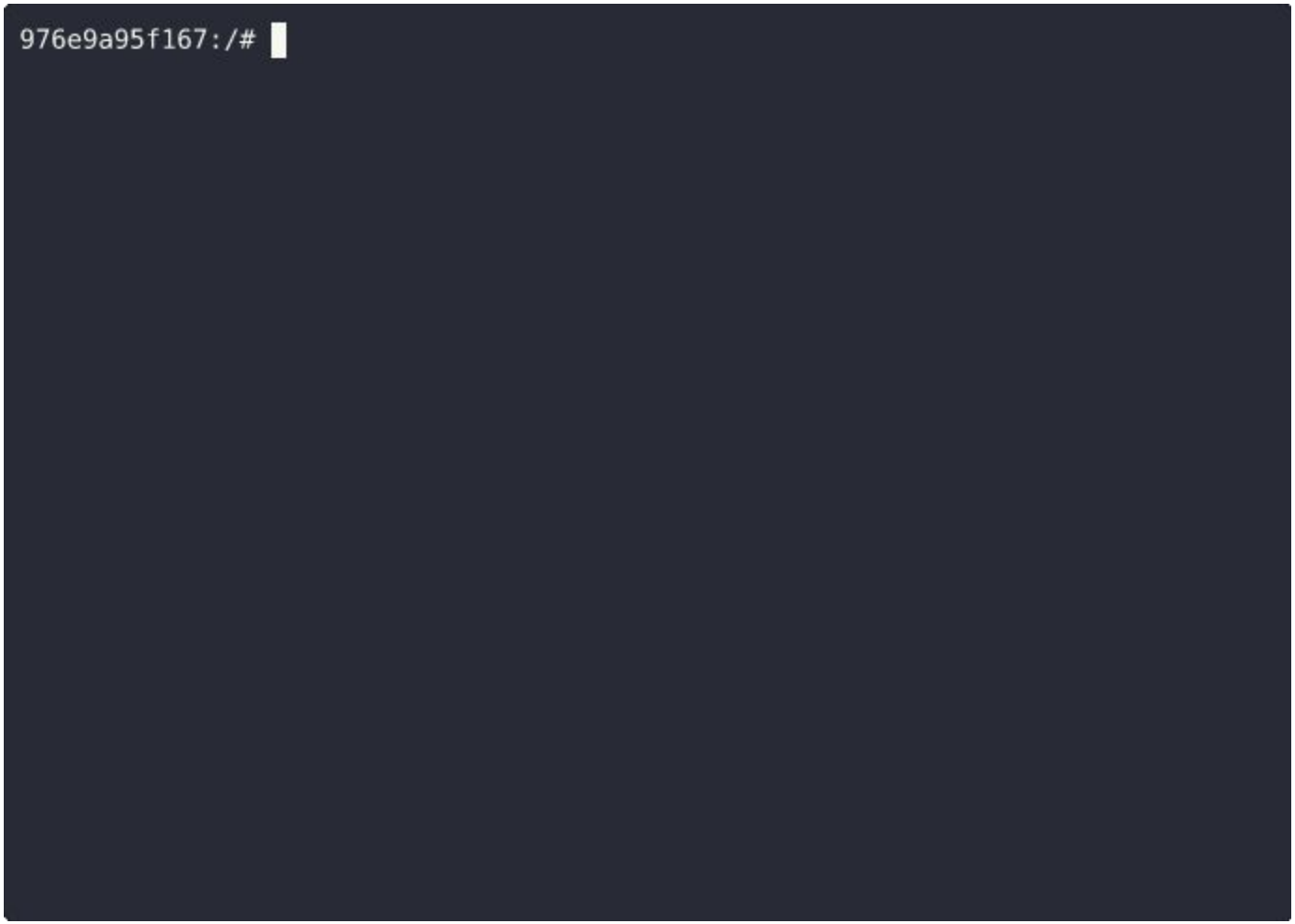
- **Driver-dependent creation-option name**

```
$ gdal raster convert --of=GTIFF --co=<TAB><TAB>
ALPHA=                          ENDIANNESS=                     JXL EFFORT=
BIGTIFF=                        GEOTIFF KEYS FLAVOR=            JXL LOSSLESS=
BLOCKXSIZE=                     GEOTIFF_VERSION=                LZMA_PRESET=
[...]
```

- **creation-option value suggestion: enumeration**

```
$ gdal raster convert --of=GTIFF --co COMPRESS=
CCITTFAX3 CCITTFAX4     CCITTRLE   DEFLATE   JPEG    JXL
LERC      LERC DEFLATE  LERC_ZSTD    LZMA      LZW     NONE
PACKBITS  WEBP          ZSTD
```

Bash completion



Python usage

- **Getting output information as JSON**

```
>>> alg = gdal.Run("raster", "info", input="byte.tif")
>>> info_as_dict = alg.Output()
```

- **Getting output information as a GDAL dataset object**

```
>>> with gdal.Run("raster reproject",
              input=src_ds, output_format="MEM",
              dst_crs="EPSG:4326"}) as alg:
>>>    values = alg.Output().ReadAsArray()
```

⇒ **See https://gdal.org/programs/gdal_cli_from_python.html**

`gdal_translate`, `ogr2ogr` vs `gdal`

- No breaking changes to main CLIs
- New capabilities -> `gdal` commands
- Migration and decommission schedule of `gdal`-ported Python-based CLIs
- No promise of subcommands, options, or GDALG compatibility until ~3.16.0
- User feedback needed on `gdal`

VRT enhancements / gdal raster calc

- $ gdal raster calc --input rgbnir.tif --output ndvi.vrt --calc "(X[4] - X[1]) / (X[1] + X[4])"

```
<VRTDataset rasterXSize="20" rasterYSize="20">
 <VRTRasterBand dataType="Float64" band="1" subClass="VRTDerivedRasterBand">
      <SimpleSource name="X[1]">
      <SourceFilename relativeToVRT="0">rgbnir.tif</SourceFilename>
      <SourceBand>1</SourceBand>
      </SimpleSource>
      <SimpleSource name="X[4]">
      <SourceFilename relativeToVRT="0">rgbnir.tif</SourceFilename>
      <SourceBand>4</SourceBand>
      </SimpleSource>
      <PixelFunctionType>expression</PixelFunctionType>
      <PixelFunctionArguments dialect="muparser" expression="(X[4] - X[1]) / (X[1] + X[4])" />
 </VRTRasterBand>
</VRTDataset>
```

Miscallenaous (1 / 2)

- gdal_viewshed enhancements:
  - Multi threaded optimization
  - Observer out of raster extent
  - Cumulative viewshed
- STAC GeoParquet in the GDAL Tile Index driver
- ADBC (Arrow Database Connectivity) driver, in particular with support for DuckDB or Parquet datasets

Miscallenaous (2 / 2)

- **LiberTIFF: an alternate native thread-safe read-only GeoTIFF reader**
- **Support for embedding resource files into the library**
- **Addition of a OGR_SCHEMA open option to selected OGR drivers (CSV, GeoJSON, GML, SQLITE)**

Thank you sponsors!

- Gold level:

- Silver level:

Safe Software

- Bronze level:

geo MAXAR Toitū Te Whenua Land Information New Zealand GeoCzech

- Supporter level:

Dynamic Graphics, Inc. Route4Me, Inc. Kaplan Open Source Consulting Myles Sutherland Satelligence Umbra Vortex f.d.c. Regrid Phoenix LiDAR Systems, LLC PIX4D T-Kartor Space Intelligence

GDAL

.... new ones welcome!