qfyaml

Release 0.4.2

Bob Yantosca

Aug 09, 2022

OVERVIEW

The **qfyaml** package allows you to read information from YAML files into Fortran code, using only the F95 standard language features. It is compatible with most versions of GNU Fortran and Intel Fortran.

CHAPTER

ONE

ABOUT QFYAML

The Quick Fortran YAML parser (**qfyaml**) is a standalone package for reading fields from YAML files directly into Fortran. It was created by Bob Yantosca as a way to read configuration files into the GEOS-Chem model.

Qfyaml (which builds upon the config-fortran package by H. J. Teunissen) uses F95 language syntax so that it can be built with most versions of the GNU and Intel Fortran compilers.

CHAPTER

TWO

SYSTEM REQUIREMENTS

You must have the following packages installed on your computer in order to use **`qfyaml`**:

1. Git, used for downloading the source code.
2. CMake, version 3.5 or later.
3. Either the GNU Fortran Compiler or the Intel Fortran Compiler.

CHAPTER

THREE

INSTALLATION AND COMPILATION

3.1 Downloading qfyaml

Use Git to clone the source code from the qfyaml repository to your computer system:

```
$ git clone https://github.com/yantosca/qfyaml.git
```

This will create a folder named `qfyaml` in your disk space.

The main **`qfyaml`** source code file is `qfyaml/src/qfyaml_mod.F90`. Several test programs are included in the `qfyaml/test` folder.

3.2 Configuring qfyaml

You may now call CMake to begin the configuration process. During configuration, CMake will first check if you have everything on your system that is required to compile the **`qfyaml`** source code. If everything checks out, then Cmake will create several Makefiles to be used during compilation.

Navigate to the `qfyaml/bin` folder:

```
$ cd qfyaml/bin
```

and call CMake with this command:

```
$ cmake .. -B ../build
```

The `-B ../build` command tells CMake to create the Makefiles in the `qfyaml/build` folder.

When the test runs, you should see output similar to this:

```
-- The Fortran compiler identification is GNU 11.1.0
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Check for working Fortran compiler: /bin/gfortran - skipped
-- Configuring done
-- Generating done
-- Build files have been written to: /home/bob/work/qfyaml/build
```

3.2.1 Compling qfyaml

Once the configuration step has completed successfully, you may compile the **qfyaml** source code.

Type the following at the command line:

```
$ make -C ../build
```

The `-C` command tells **make** to look in the `../build` folder for the Makefiles that were created by CMake. These Makefiles will direct the build process.

Important: You will need to repeat the compilation process each time you modify `qfyaml_mod.F90` or one of the `test_*.F90` source code files.

When the test runs, you should see output similar to this.

```
make: Entering directory '/home/bob/work/qfyaml/build'
make[1]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
Scanning dependencies of target QfYaml
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
[  9%] Building Fortran object src/CMakeFiles/QfYaml.dir/qfyaml_mod.F90.o
[ 18%] Linking Fortran static library libQfYaml.a
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 18%] Built target QfYaml
make[2]: Entering directory '/home/bob/work/qfyaml/build'
Scanning dependencies of target Common
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
[ 27%] Building Fortran object test/CMakeFiles/Common.dir/precision_mod.F90.o
[ 36%] Building Fortran object test/CMakeFiles/Common.dir/roundoff_mod.F90.o
[ 45%] Linking Fortran static library libCommon.a
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 45%] Built target Common
make[2]: Entering directory '/home/bob/work/qfyaml/build'
Scanning dependencies of target test_qfyaml.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
[ 54%] Building Fortran object test/CMakeFiles/test_qfyaml.x.dir/test_qfyaml.F90.o
[ 63%] Linking Fortran executable test_qfyaml.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 63%] Built target test_qfyaml.x
make[2]: Entering directory '/home/bob/work/qfyaml/build'
Scanning dependencies of target test_species_database.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
[ 72%] Building Fortran object test/CMakeFiles/test_species_database.x.dir/test_
↪species_database.F90.o
[ 81%] Linking Fortran executable test_species_database.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 81%] Built target test_species_database.x
make[2]: Entering directory '/home/bob/work/qfyaml/build'
Scanning dependencies of target test_geoschem_config.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
```

(continues on next page)

(continued from previous page)

```
[ 90%] Building Fortran object test/CMakeFiles/test_geoschem_config.x.dir/test_
↪geoschem_config.F90.o
[100%] Linking Fortran executable test_geoschem_config.x
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[100%] Built target test_geoschem_config.x
make[1]: Leaving directory '/home/bob/work/qfyaml/build'
make: Leaving directory '/home/bob/work/qfyaml/build'
```

3.3 Installing qfyaml

Once compilation has finished successfully, you may install the compiled code (and various input files) to the `qfyaml/bin` folder.

Type at the command line:

```
$ make -C ../build install
```

Important: You will need to repeat the installation process each time you modify `qfyaml_mod.F90` or one of the `test_*.F90` source code files.

When the test runs, you should see output similar to this:

```
make: Entering directory '/home/bob/work/qfyaml/build'
make[1]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 18%] Built target QfYaml
make[2]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 45%] Built target Common
make[2]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 63%] Built target test_qfyaml.x
make[2]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[ 81%] Built target test_species_database.x
make[2]: Entering directory '/home/bob/work/qfyaml/build'
make[2]: Leaving directory '/home/bob/work/qfyaml/build'
[100%] Built target test_geoschem_config.x
make[1]: Leaving directory '/home/bob/work/qfyaml/build'
Install the project...
-- Install configuration: ""
-- Installing: /home/bob/work/qfyaml/bin/test_qfyaml.x
-- Up-to-date: /home/bob/work/qfyaml/bin/input.yml
-- Installing: /home/bob/work/qfyaml/bin/test_species_database.x
-- Up-to-date: /home/bob/work/qfyaml/bin/species_database.yml
-- Up-to-date: /home/bob/work/qfyaml/bin/species_database_apm.yml
-- Up-to-date: /home/bob/work/qfyaml/bin/species_database_tomas.yml
-- Installing: /home/bob/work/qfyaml/bin/test_geoschem_config.x
-- Up-to-date: /home/bob/work/qfyaml/bin/geoschem_config.yml
make: Leaving directory '/home/bob/work/qfyaml/build'
```

Then to see the files that were installed, type:

```
$ cd ../bin
$ ls -1
```

and you will see this directory listing:

```
geoschem_config.yml
input.yml
species_database_apm.yml
species_database_tomas.yml
species_database.yml
test_geoschem_config.x
test_qfyaml.x
test_species_database.x
```

Executable files for the various tests end with the `.x` extension. The corresponding configuration files in YAML format end with the `.yml` format.

3.4 Removing files created during compilation

To remove all files in the `bin/` and `build/` folders, type:

```
$ cd bin
$ ./cleanup.sh
```

CHAPTER

FOUR

RUNNING THE QFYAML TEST PROGRAMS

Several test programs are included with **qfyaml**. After installation, executables for these tests will be placed in the `qfyaml/bin` folder.

4.1 General test programs

The following test programs can be used to help debug issues with the **qfyaml** code itself.

4.1.1 test_qfyaml.x

This test reads a general YAML file (input.yml) that is designed to catch several issues with parsing YAML markuup.

To run this test, type at the command line:

```
$ cd qfyaml/bin      # Skip if you are already in qfyaml/bin
$ ./test_qfyaml.x
```

This test will parse the `input.yml` file and echo back the output. If the test is successful you will see this output:

```
### Reading qfyaml.yml

### YAML VARIABLES
author%age                          :      29
author%fav_reals                    :    1.00   2.00
author%more_reals                   :    3.141590   2.781280   8.573900 101.324997
author%lots_of_work                 :       F
author_name%first                   : Homer
author_name%full                    : Homer J. Simpson
filename                            : another_file
weather%humidity                    :     99.858582
weather%temperature%daily           :     23.043436
weather%temperature%weekly%units    : K
weather%pressure                    :   1013.250000

### FIND NEXT-HIGHER VARIABLES IN "weather"
            1 weather%humidity
            2 weather%pressure
            3 weather%temperature

### YAML SEQUENCES
fruits
           1 Apples
```

(continues on next page)

(continued from previous page)

```
           2 Bananas
           3 Oranges

more_fruits%p_fruits
           1 Pears
           2 Plums
           3 Peaches
           4 Pomegranites

even_more_fruits%exotic_fruits%hard_to_find
           1 Kumquats
           2 Kiwi
           3 Passion_fruit
           4 Star_fruit
           5 Durians

#### finishing
```

4.2 GEOS-Chem-specific test programs

The following test programs can be used to debug source code for reading YAML-format configuration files into the GEOS-Chem model.

4.2.1 test_config.x

This test program attempts to read the GEOS-Chem master configuration file and echo back output. The master configuration file will replace `input.geos` in GEOS-Chem 14.0.0 and later.

To run this test, type at the command line:

```
$ cd qfyaml/bin      # Skip if you are already in qfyaml/bin
$ ./test_config.x
```

And you should see output such as:

```
### Reading input_options.yml
 simulation%start
 ==>     20190701           0
 simulation%end
 ==>     20190801           0
 simulation%data_dir
 ==> /n/holyscratch01/external_repos/GEOS-CHEM/gcgrid/data/ExtData
 simulation%met_field
 ==> MERRA2
 simulation%name
 ==> fullchem
 simulation%species_database_file
 ==> species_database.yml
 simulation%debug_printout
 ==>  F
 simulation%use_gcclassic_timers
 ==>  F
```

(continues on next page)

(continued from previous page)

```
grid%resolution
==> 0.5x0.625
grid%longitude_range
==>   -140.000000      -40.0000000
grid%center_lon_at_180
==>  T
grid%latitude_range
==>   -10.0000000       70.0000000
grid%half_size_polar_boxes
==>  T
grid%number_of_levels
==>           72
grid%nested_grid_simulation
==>  T
grid%buffer_zone_NSEW
==>            3           3           3           3

. . . etc . . .
```

4.2.2 test_species_database.x

This test program attemps to read the GEOS-Chem species database file and echo back output.

To run this test, type at the command line:

```
$ cd qfyaml/bin      # Skip if you are already in qfyaml/bin
$ ./test_species_database.x
```

You should see output similar to this:

```
### Reading species_database.yml
### Reading species_database_tomas.yml
            ACTA%Background_VV |    -999.00
            ACTA%DD_AeroDryDep |          F
            ACTA%DD_DustDryDep |          F
            ACTA%DD_DvzAerSnow |    -999.00
             ACTA%DD_DvzMinVal |    -999.00   -999.00
                    ACTA%DD_F0 |       1.00
                 ACTA%DD_Hstar |    4100.00
                  ACTA%Density |    -999.00
                  ACTA%Formula | CH3C(O)OH
                 ACTA%FullName | Acetic acid
               ACTA%Is_Advected |          T
                ACTA%Is_Aerosol |          F
                 ACTA%Is_DryAlt |          F
                 ACTA%Is_DryDep |          T
           ACTA%Is_HygroGrowth |          F
                   ACTA%Is_Gas |          T
                   ACTA%Is_Hg0 |          F
                   ACTA%Is_Hg2 |          F
                   ACTA%Is_HgP |          F
            ACTA%Is_Photolysis |          F
                ACTA%Is_WetDep |          T
                 ACTA%Henry_CR |    6200.00
                 ACTA%Henry_K0 |    4050.00
```

(continues on next page)

(continued from previous page)

```
                ACTA%Henry_pKa |    -999.00
            ACTA%MP_SizeResAer |           F
            ACTA%MP_SizeResNum |           F
               ACTA%MolecRatio |        1.00
                     ACTA%MW_g |       60.06
            ACTA%WD_AerScavEff |    -999.00
             ACTA%WD_CoarseAer |           F
            ACTA%WD_ConvFacI2G |    -999.00
            ACTA%WD_KcScaleFac |    -999.00   -999.00   -999.00
        ACTA%WD_KcScaleFac_Luo |    -999.00   -999.00   -999.00
             ACTA%WD_LiqAndGas |           F
            ACTA%WD_RainoutEff |    -999.00   -999.00   -999.00
        ACTA%WD_RainoutEff_Luo |    -999.00   -999.00   -999.00
             ACTA%WD_RetFactor |        0.02

. . . etc . . .
```

CHAPTER

FIVE

KNOWN BUGS

This page lists known bugs in **qfyaml**. See the Github issues page for updates on their status.

5.1 Version 0.3.2

5.1.1 Error parsing categories

We discovered an error parsing this YAML file, where the `wet_deposition` tag is more than 2 indentation levels behind behind the previous line.

```
operations:
  transport:
    passive_species:
      CH3ITracer:
        long_name: Methyl_iodide
        mol_wt_in_g: 142.0
        lifetime_in_s: 4.32e5
        default_bkg_conc_in_vv: 1.0e-20
  wet_deposition:
    activate: true
```

This has now been fixed in qfyaml 0.3.3.

NOTE: For best results with qfyaml, we recommend formatting YAML files so that they contain a consistent indentation level throughout the file (i.e. such as 2 or 4 spaces). Editors such as Emacs can do this easily.

5.2 Version 0.3.0

5.2.1 Error parsing YAML sequence items containing spaces

Items in YAML sequences containing spaces are not parsed properly. For example:

```
hard_to_find_fruits:
  - Passion fruit
  - Star fruit
```

will be parsed as 4 separate items (`Passion`, `fruit`, `Star`, and `fruit`) instead the expected 2 items `Passion fruit` and `Star fruit`. We will fix this in a future version.

5.2.2 Workaround

Use underscores instead of spaces in YAML sequence items:

```
hard_to_find_fruits:
- Passion_fruit
- Star_fruit
```

And then you can remove the underscores in post-processing.

5.2.3 Error returning long YAML sequences

It was discovered that only a subset long YAML sequences were being returned. Upon further investigation, a string variable in routine `Get_Fields_string` was found to be too short to hold all of the stored data within a YAML variable.

We have fixed this behavior in **`qfyaml 0.3.1`**.

Arrays passed to QFYAML_Add_Get had to be the same size as the data in the YAML file ————————————

When passing an array to routine `QFYAML_Add_Get`, an error would be returned if the array was not the same length as the array or sequence in the YAML file.

In **`qfyaml 0.3.1`**, arrays that are larger than the size of the data may be passed to `QFYAML_Add_Get`. This will let you declare an array size that is sufficiently large in the calling routine. This can be especially useful if you do not know the size of the data to be read in from the YAML file.

CHAPTER

SIX

EDITING THESE DOCS

This documentation is generated with Sphinx. This page describes how to contribute to the GCPy documentation.

6.1 Quick start

You need the Sphinx Python to build (and therefore edit) this documentation. Assuming you already have Python installed, install Sphinx:

```
$ pip install sphinx
```

To build the documentation, navigate to `qfyaml/docs` and make the html target:

```
$ cd qfyaml/docs
$ make html
```

This will generate the HTML documentation in `qfyaml/docs/build/html` from the reST files in `qfyaml/docs/source`. You can view this local HTML documentation by opening `index.html` in your web-browser.

Note: You can clean the documentation with `make clean`.

6.2 Learning reST

Writing reST can be a bit tricky at first. Whitespace matters (just like in Python), and some directives can be easily miswritten. Two important things you should know right away are:

- Indents are 3-spaces
- "Things" are separated by 1 blank line (e.g., a list or code-block following a paragraph should be separated from the paragraph by 1 blank line)

You should keep these in mind when you're first getting started. Dedicating an hour to learning reST will save you time in the long-run. Below are some good resources for learning reST.

- reStructuredText primer: (single best resource; however, it's better read than skimmed)
- Official reStructuredText reference (there is *a lot* of information here)
- Presentation by Eric Holscher (co-founder of Read The Docs) at DjangoCon US 2015 (the entire presentation is good, but reST is described from 9:03 to 21:04)
- YouTube tutorial by Audrey Tavares's

A good starting point would be Eric Holscher's presentations followed by reading the reStructuredText primer.

6.3 Style guidelines

Important: This documentation is written in semantic markup. This is important so that the documentation remains maintainable by the GEOS-Chem Support Team. Before contributing to this documentation, please review our style guidelines. When editing the documentation, please refrain from using elements with the wrong semantic meaning for aesthetic reasons. Aesthetic issues should be addressed by changes to the theme (not changes to reST files).

For **titles and headers**:

- H1 titles should be underlined by `#` characters
- H2 headers should be underlined by `-` characters
- H3 headers should be underlined by `^` characters
- H4 headers should be avoided, but if necessary, they should be underlined by `"` characters

File paths occuring in the text should use the `:literal:` role.

Inline code, or references to variables in code, occuring in the text should use the `:code:` role.

Code snippets should use `.. code-block:: <language>` directive like so

```
.. code-block:: python

   import gcpy
   print("hello world")
```

The language can be "none" to omit syntax highlighting.

For command line instructions, the "console" language should be used. The `$` should be used to denote the console's prompt. If the current working directory is relevant to the instructions, a prompt like `$USER:~/path1/path2$` should be used.

Inline literals (such as the `$` above) should use the `:literal:` role.