ecCodes

GRIB Fortran 90 - Python APIs
Part 2

Dominique Lucas and Xavi Abellan

Dominique.Lucas@ecmwf.int Xavier.Abellan@ecmwf.int

ECMWF

Content

- Remember versus

- Indexed access to GRIB data

- Encoding a loaded GRIB message

- Python API

Example – codes_get

```fortran
! Load all the GRIB messages contained in file.grib1
call codes_open_file(ifile, 'file.grib1','r')
n=1
call  codes_grib_new_from_file(ifile,igrib(n), iret)
LOOP: do while (iret /= CODES_END_OF_FILE)
    n=n+1; call codes_grib_new_from_file(ifile,igrib(n), iret)
end do LOOP
```

Loop on all the messages in a file. A new grib message is loaded from file. igrib(n) is the grib id to be used in subsequent calls

```fortran
! Decode/encode data from the loaded message
read*, indx                            ! Choose one grib loaded GRIB message to decode
call codes_get( igrib(indx) , "dataDate", date)
call codes_get(igrib(indx), "typeOfLevel", typeOfLevel)
call codes_get(igrib(indx), "level", level)
call codes_get_size(igrib(indx), "values", nb_values); allocate(values(nb_values))
call codes_get(igrib(indx), "values", values)
print*, date, levelType, level, values(1), values(nb_values)
```

Values is declared as real, dimension(:), allocatable:: values

```fortran
! Release
do i=1,n
    call codes_release(igrib(i))
end do
deallocate(values)
call codes_close_file(ifile)
```

ECMWF

Example – grib_get

```fortran
! Load all the GRIB messages contained in file.grib1
call grib_open_file(ifile, 'file.grib1','r')
n=1
call  grib_new_from_file(ifile,igrib(n), iret)
LOOP: do while (iret /= GRIB_END_OF_FILE)
    n=n+1; call grib_new_from_file(ifile,igrib(n), iret)
end do LOOP
```

Loop on all the messages in a file.
A new grib message is loaded
from file. igrib(n) is the grib id to
be used in subsequent calls

```fortran
! Decode/encode data from the loaded message
read*, indx                          ! Choose one grib loaded GRIB message to decode
call grib_get( igrib(indx) , "dataDate", date)
call grib_get(igrib(indx), "typeOfLevel", typeOfLevel)
call grib_get(igrib(indx), "level", level)
call grib_get_size(igrib(indx), "values", nb_values); allocate(values(nb_values))
call grib_get(igrib(indx), "values", values)
print*, date, levelType, level, values(1), values(nb_values)
```

Values is declared as
real, dimension(:), allocatable:: values

```fortran
! Release
do i=1,n
    call grib_release(igrib(i))
end do
deallocate(values)
call grib_close_file(ifile)
```

ecCodes indexed access

- Several subroutines:

codes_index_create(indexid, filename, keys, status) or grib_index_create

to create the index of the content of a file

codes_index_get_size(indexid, key, size, status)   or grib_index_get_size

to get the dimension of a key in the index

codes_index_get(indexid, key, values, status)   or grib_index_get

to get the different "values" for a key in the index

codes_index_select(indexid, key, value, status)   or grib_index_select

to select a "value" for a key in the index

ecCodes indexed access

- Several subroutines:

    codes_new_from_index(indexid, igrib, status)   or grib_new_from_index

    to load the GRIB message corresponding to the selection made.

    codes_index_release(indexid, status)   or grib_index_release

    to release the index.

    and … codes_release(igrib)   or grib_release

    to release the GRIB message.

- Indexed access is usually much faster than sequential access for "random" access.

Example – indexed access

```fortran
! create an index from a grib file using two keys
call codes_index_create(idx,'ensemble.grib','paramId')

! get the number of distinct values of parameters in the index
call codes_index_get_size(idx,'paramId',paramIdSize)
! allocate the array to contain the list of distinct paramId
allocate(paramId(paramIdSize))
! get the list of distinct parameters from the index
call codes_index_get(idx,'paramId',paramId)
```

List of keys to be indexed, comma separated, without any spaces, between one single set of quotes.

File "ensemble.grib" contains all ensemble members for several parameters.

```fortran
count=1
do i=1,paramIdSize ! loop on paramId
   ! select paramId=paramId(i)
   call codes_index_select(idx,'paramId',paramId(i))
   call codes_new_from_index(idx,igrib,iret)
```

Note that I have to select a value for all the keys used to build the index.

I load the first grib message I need into memory.

Example – indexed access

Note that several grib messages may be available for one selection of my index, therefore this loop.

```fortran
do while (iret /= GRIB_END_OF_INDEX)
   call codes_is_missing(igrib,'number', is_missing);
   if (is_missing /= 1) then
      call codes_get(igrib,'number',onumber)
   else
      onumber=-9999
   end if
   call codes_get(igrib,'level',olevel)
   print*,'param:'', paramld(i),' level:',olevel, ' number:',onumber
   call codes_release(igrib)
   call codes_new_from_index(idx,igrib,iret)
end do


end do ! loop on paramld
call codes_index_release(idx)
```

ecCodes indexed access – i/o

- An index can be saved into a file, to be re-used.

  codes_index_write(indexid, filename, status)   or grib_index_write
  to save an index to a file

  codes_index_read(indexid, filename, status)   or grib_index_read
  to load an index file previously created with codes_index_write

- One can also add the content of a data file to an index.

  codes_index_add_file(indexid, filename, status)   or grib_index_add_file
  to add the content of a data file to an index.

- One can build an index with the ecCodes command grib_index_build.

- The command 'grib_dump –D <index_file>' will show the content of an index file.

- A little more on this in the practical session.

Encoding a loaded GRIB message

- The idea is to "encode" as little as possible! You will never "encode" the whole GRIB message.

- one main subroutine to "encode":

    codes_set(igrib, keyname, values, status)   or grib_set

    integer, intent(in)   :: igrib

    character(len=), intent(in)   :: keyname*

    <type>,[dimension(:),] intent(in) :: values

    integer, optional, intent(out)   :: status

    Where <type> is integer or single/double real precision or string

    Input arguments
    Output arguments

- Writing a message:

    call codes_write(igrib, output_file)   or grib_write

    Note that a grib message written with codes_write will be syntactically correct, but it may be semantically incorrect.

ECMWF

Creation of a new message

- A new message can be created from a sample:

  - A sample is an example grib message available in the sample directory. The default sample directory can be found with the command 'codes_info'. Sample file names end up with a suffix '.tmpl'. You can create your own samples and change/add the environment variable ECCODES_SAMPLES_PATH to point to them.

  - Creating a new grib message from a sample:

  call codes_grib_new_from_samples(igrib, samplename, status) or grib_new_from_samples

- A new message can be cloned (copied) from another message

  call codes_clone(igrib_src,igrib_dest,status)   or grib_clone

Input arguments

Output arguments

Example – codes_set

```fortran
!  STEP-1: open output file and load a GRIB message from a sample "GRIB1"
call codes_open_file(outfile, 'out.grib1','w')          ! GRIB1.tmpl is a GRIB-1 file located
call codes_grib_new_from_samples(igrib, "GRIB1")        ! in the samples directory


!  STEP-2: Get some information from the loaded message
call codes_get_size(igrib, "values", nb_values)
allocate(values(nb_values))                             ! Declared as real, dimension(:), allocatable
call model(values); values(1:100) = 9999.0             ! Compute values and set some missing values

! STEP-3: set the new GRIB message
call codes_set(igrib,'missingValues', 9999.0)           ! Tells the GRIB-API 9999.0 is the missing value
call codes_set(igrib,'bitmapPresent', 1)
call codes_set(igrib,"values", values)                  ! Set values as 1D real array of size nb_values

! STEP-4: write modified message to a file
call codes_write(igrib,outfile)
call codes_release(igrib)
call codes_close_file(outfile)
deallocate(values)
```

Changing grid definition and packing type

- You can apply a grid definition or change the packing type by changing the keys gridType and/or packingType, e.g:

    call codes_set(igrib,'gridType', 'polar_stereographic')   or grib_set

  will define a "Polar Stereographic Projection Grid" for your message.

    call codes_set(igrib,'packingType', 'grid_simple')   or grib_set

  will pack the data as simple packing.

- The grid definitions and grib packing types are listed under:

    https://software.ecmwf.int/wiki/display/ECC/GRIB%3A+Keys

Usage different packing types

- GRIB data can be packed in different ways, e.g. simple packing, second order packing, …

- Not all packing types are available for GRIB1 and GRIB2.

- A packing type will be available either for grid-point or spectral field.

- The type of packing used will affect the size of your GRIB messages produced, e.g. second order packing may produce messages twice as small as simple packing.

- The type of packing used will affect the time it takes to pack/unpack your data, e.g. second order packing may be significantly slower than simple packing.

- Packing doesn't lose information.

- More on this in the practical session …

Python API – Indexing

iid = *codes_index_new_from_file* (file, keys) Returns a handle to the created index	*grib_index_new_from_file*
codes_index_add_file (iid, file) Adds a file to an index.	*grib_index_add_file*
codes_index_write (iid, file) Writes an index to a file for later reuse.	*grib_index_write*
iid = *codes_index_read* (file) Loads an index saved with *codes_index_write* to a file.	*grib_index_read*
codes_index_release (iid) Release the index	*grib_index_release*

Python API – Indexing

size = *codes_index_get_size* (iid, key) Gets the number of distinct values for the index key.	*grib_index_get_size*
values = *codes_index_get* (iid, key, ktype=str) Gets the distinct values of an index key.	*grib_index_get*
codes_index_select (iid, key, value) Selects the message subset with key==value.	*grib_index_select*

gid = *codes_new_from_index* (iid)
 	
    Same as *codes_grib_new_from_file*
 	
    Release with *codes_release*(gid)
	grib_new_from_index

Python API – Encoding

codes_set (gid, key, value) Sets the value for a scalar key in a grib message.	**grib_set**
codes_set_array (gid, key, value) Sets the value for an array key in a grib message. The input array can be a numpy.ndarray or a Python sequence like tuple, list, array, ...	**grib_set_array**
codes_set_values (gid, values) Utility function to set the contents of the 'values' key.	**grib_set_values**
clone_id = **codes_clone** (gid_src) Creates a copy of a message. You can directly write to file with *codes_write* Don't forget to *codes_release*	**grib_clone**

References

- GRIB-1, GRIB-2:

  http://www.wmo.int/pages/prog/www/WMOCodes.html

- ecCodes:

  https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home

- ecCodes Fortran, C or Python interfaces to GRIB data (currently on the GRIB API wiki):

  https://software.ecmwf.int/wiki/display/ECC/ecCodes+API+Reference

- Examples:

  https://software.ecmwf.int/wiki/display/ECC/GRIB+examples

- GRIBEX – GRIB API conversion:

  https://software.ecmwf.int/wiki/display/GRIB/GRIBEX+keys