2020 Summer School on Effective HPC for Climate and Weather

Input/Output and Middleware

Luciana Pedro, Julian Kunkel

Department of Computer Science, University of Reading

18 June 2020
Outline

1. NetCDF Files and C
2. NetCDF Utilities
3. Practising

Disclaimer: This material reflects only the author’s view and the EU-Commission is not responsible for any use that may be made of the information it contains
Learning Objectives

- Execute programs in C that read and write NetCDF files in a metadata-aware manner
- Analyze, manipulate and visualise NetCDF data
The files and data used in this presentation were collected on the Unidata website.

- https://www.unidata.ucar.edu/

All files used here are available in the following Git Repository:

- https://github.com/ESiWACE/io-training

These files are also available with the NetCDF main installation, in the directory examples.

For more information about how to install NetCDF in your personal computer, from scratch, check Section 4.
File Reference: simple_xy_wr.c

- This is an example program demonstrating a simple 2D write. It is intended to illustrate the use of the NetCDF C API.
  
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__wr_8c.html
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__wr_8c_source.html

- Dependency graph for simple_xy_wr:
File `simple_xy_wr.c`: Header and Constants Declaration

```c
#include <stdlib.h>
#include <stdio.h>
#include <netcdf.h>

/* This is the name of the data file we will create. */
#define FILE_NAME "simple_xy.nc"

/* We are writing 2D data, a 6 x 12 grid. */
#define NDIMS 2
#define NX 6
#define NY 12

/* Handle errors by printing an error message and exiting with a non-zero status. */
#define ERRCODE 2
#define ERR(e) {printf("Error: %s\n", nc_strerror(e)); exit(ERRCODE);}
```

- Standard C libraries and main NetCDF library.
- Define the name for the NetCDF file.
- Define the total number of dimensions.
- Define each of the dimensions.
- Define error codes and messages.
int main()
{
    /* When we create NetCDF variables and dimensions, we get back an 
     * ID for each one. */
    int ncid, x_dimid, y_dimid, varid;
    int dimids[NDIMS];

    /* This is the data array we will write. It will be filled with a 
     * progression of numbers for this example. */
    int data_out[NX][NY];

    /* Loop indexes, and error handling. */
    int x, y, retval;

    ...  
   ...  
   ...
}
File `simple_xy_wr.c`: Creating (loading!) Data

```c
int main()
{
    ...

    /* Create some pretend data. If this wasn't an example program, we
     * would have some real data to write, for example, model
     * output. */
    for (x = 0; x < NX; x++)
        for (y = 0; y < NY; y++)
            data_out[x][y] = x * NY + y;

    ...
}
```

- Real data can be loaded using different databases and functions in C.
- In this example, the data is generated by a simple formula.
File `simple_xy_wr.c`: Creating the NetCDF file

```c
... int main() {
... /* Always check the return code of every NetCDF function call. In
* this example program, any retval which is not equal to NC_NOERR
* (0) will cause the program to print an error message and exit
* with a non-zero return code. */

/* Create the file. The NC_CLOBBER parameter tells NetCDF to
* overwrite this file, if it already exists. */
if ((retval = nc_create(FILE_NAME, NC_CLOBBER, &ncid))
  ERR(retval);
... }
```

- The function to create a NetCDF file is called `nc_create`.

- This function has three parameters:
  - The name of the file.
  - The mode to open the file.
  - It returns the id for the NetCDF file.
File `simple_xy_wr.c`: Defining the Dimensions

```c
... int main() {
  ...

  /* Define the dimensions. NetCDF will hand back an ID for each. */
  if ((retval = nc_def_dim(ncid, "x", NX, &x_dimid)))
    ERR(retval);
  if ((retval = nc_def_dim(ncid, "y", NY, &y_dimid)))
    ERR(retval);

  /* The dimids array is used to pass the IDs of the dimensions of
   * the variable. */
  dimids[0] = x_dimid;
  dimids[1] = y_dimid;
  ...
}
```

- The function to define new dimensions in a NetCDF file is called `nc_def_dim`.

- This function has four parameters:
  - The id of the NetCDF file.
  - The name of the dimension to be created.
  - The size of the dimension to be created.
  - It returns the id for created dimension.

- The vector `dimids` stores the ids for the created dimensions.
File `simple_xy_wr.c`: Defining a Variable

```c
int main()
{
  ...
  /* Define the variable. The type of the variable in this case is
   * NC_INT (4-byte integer). */
  if ((retval = nc_def_var(ncid, "data", NC_INT, NDIMS, 
      dimids, &varid)))
    ERR(retval);

  ...
}
```

- The function to define new variables in a NetCDF file is called `nc_def_var`.

- This function has six parameters:
  - The id of the NetCDF file.
  - The name of the variable to be created.
  - The type of the variable to be created.
  - The number of dimensions of the variable.
  - The vector that stores the ids for the dimensions.
  - It returns the id for created variable.
File `simple_xy_wr.c`: Defining a Variable

```c
...  

int main()
{
    ...

    /* End define mode. This tells NetCDF we are done defining
     * metadata. */
    if ((retval = nc_enddef(ncid)))
        ERR(retval);

    ...
}
```

- **Classic NetCDF:**
  - **In define mode,** dimensions, variables, and new attributes can be created but variable data cannot be read or written.
  - **In data mode,** data can be read or written and attributes can be changed, but new dimensions, variables, and attributes cannot be created.

- **NOTE:** NetCDF-4 does not distinguish between define and data modes.
File simple_xy_wr.c: Writing Data into the File

```c
...  
int main()
{
    ...  

    /* Write the pretend data to the file. Although NetCDF supports  
        reading and writing subsets of data, in this case we write all  
        the data in one operation. */  
    if ((retval = nc_put_var_int(ncid, varid, &data_out[0][0])))
        ERR(retval);
    ...  
}
```

- The function to write the data in a variable is called `nc_put_var_*`. In this example, we have `nc_put_var_int`.

- This function has four parameters:
  - The id of the NetCDF file.
  - The id of the variable that will store the data.
  - (A pointer to) the data.
File simple_xy_wr.c: Writing Data into the File

```c
... 
int main()
{
    ...

    /* Close the file. This frees up any internal NetCDF resources associated with the file, and flushes any buffers. */
    if ((retval = nc_close(ncid)))
        ERR(retval);

    ...
}
```

- The function to close a NetCDF file is called `nc_close`.

- This function has one parameter:
  - The id of the NetCDF file.
File simple_xy_wr.c: Getting SUCCESS!

```c
...
int main()
{
...
    printf("*** SUCCESS writing example file simple_xy.nc!\n");
    return 0;
}
```

- If everything is done properly, we end the main program with a nice and encouraging message.
- Hopefully, also with a new NetCDF file!
Using nc-config

The nc-config command-line utility assists with the setting of compiler and linker flags for building applications.

nc-config is a simple script that reports the configuration flags used during the NetCDF build, as well as the installed version of the NetCDF C-based libraries.

It has lots of options, listed by invoking $(nc-config --all).

Here we will use nc-config to compile and link a C application:

```bash
gcc myapp.c -o myapp $(nc-config --libs --cflags)
```
Compiling and Running the File simple_xy_wr.c

- Create (copy!) and compile the file simple_xy_wr.c.
  ```
  gcc simple_xy_wr.c -o simple_xy_wr $(nc-config --libs --cflags)
  ```

- Run the file simple_xy_wr.
  ```
  ./simple_xy_wr
  *** SUCCESS writing example file simple_xy.nc!
  ```

- Check that the file simple_xy.nc is in your directory.
File Reference: simple_xy_rd.c

- This is a simple example which reads a small dummy array that was written by simple_xy_wr.c.
  - [https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__rd_8c.html](https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__rd_8c.html)
  - [https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__rd_8c_source.html](https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__rd_8c_source.html)

- Dependency graph for simple_xy_rd:
int main()
{
    /* Open the file. NC_NOWRITE tells NetCDF we want read-only access * to the file. */
    if ((retval = nc_open(FILE_NAME, NC_NOWRITE, &ncid)))
        ERR(retval);

    /* Get the varid of the data variable, based on its name. */
    if ((retval = nc_inq_varid(ncid, "data", &varid)))
        ERR(retval);

    /* Read the data. */
    if ((retval = nc_get_var_int(ncid, varid, &data_in[0][0])))
        ERR(retval);

    /* Check the data. */
    for (x = 0; x < NX; x++)
        for (y = 0; y < NY; y++)
            if (data_in[x][y] != x * NY + y)
                return ERRCODE;

    /* Close the file, freeing all resources. */
    if ((retval = nc_close(ncid)))
        ERR(retval);
}
Reading the File simple_xy.nc

- Check that the file simple_xy.nc is in your directory.

- Create (copy!), compile and run the file simple_xy_rd.c.
  - gcc simple_xy_rd.c -o simple_xy_rd $(nc-config --libs --cflags)

- Run the file simple_xy_rd.
  - ./simple_xy_rd
  - *** SUCCESS reading example file simple_xy.nc!
ncdump and ncread

- ncdump and ncread are inverses:

- Used together, ncdump and ncread can accomplish simple NetCDF manipulations with little or no programming.
**Editing a NetCDF File**

To edit metadata or data in a NetCDF file:

- Use `ncdump` to convert NetCDF file to CDL.
- Use a text editor to make desired change to CDL.
- Use `ncgen` to turn modified CDL back into NetCDF file.

**Note:** This option is not practical for huge NetCDF files or if one intends to modify lots of files. For that, need to write a program using NetCDF library.
Creating a NetCDF File

To create a new NetCDF file with lots of metadata:

- Use a text editor to write a CDL file with lots of metadata but little or no data.
- Use `ncgen` to generate corresponding C or Fortran program for writing NetCDF.
- Insert appropriate NetCDF `var_put` calls for writing data.
- Compile and run program to create NetCDF file.
- Use `ncdump` to verify result.
Using `ncdump`

- Inspect the file `simple_xy.nc` using `ncdump`:
  
  ```
  ncdump simple_xy.nc
  ```

- Inspect the metadata of the file `simple_xy.nc` using `ncdump`:
  
  ```
  ncdump -h simple_xy.nc
  ```

- Check other options for `ncdump` with:
  
  ```
  ncdump --help
  ```
NetCDF CDL Format

```c
netcdf simple_xy {
  dimensions:
  x = 6 ;
  y = 12 ;
  variables:
  int data(x, y) ;
  data:
    data =
      0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
      12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
      24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
      36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
      48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
      60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71 ;
}
```
Using `ncgen`

- Create a NetCDF file using `ncgen` and the CDL output
  - `ncdump simple_xy.nc > simple_xy_test.cdl`
  - `more simple_xy_test.cdl`
  - `ncgen -b simple_xy_test.cdl`
  - `cmp simple_xy_test.nc simple_xy.nc`
Creating the C File

- Create a C file using `ncgen` and the CDL output
  
  ```
  ngen -lc simple_xy_test.cdl > simple_xy_test.c
  
  more simple_xy_test.c
  ```

- What is the difference between the files `simple_xy_test.c` and `simple_xy_wr.c`?
  
  ```
  cmp simple_xy_test.c simple_xy_wr.c
  
  meld simple_xy_test.c simple_xy_wr.c
  ```
Starting All Over Again!

- gcc simple_xy_test.c -o simple_xy_test $(nc-config --libs --cflags)
- mv simple_xy_test.nc simple_xy_test2.nc
- ./simple_xy_test
- cmp simple_xy_test.nc simple_xy_test2.nc
Using ncview

ncview simple_xy.nc

```c
netcdf simple_xy {
  dimensions:
  x = 6 ;
  y = 12 ;
  variables:
  int data(x, y) ;
  data:
      data =
          0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
          12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
          24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
          36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
          48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
          60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71 ;
}
```
Using `ncview` – A Global Example

File `elevations.nc`
Global Potential Vegetation Dataset

- File `vegtype_5min.nc` – NetCDF 5 min data
- File `vegtype_0.5.nc` – NetCDF data aggregated to a 0.5 deg resolution

Northern Hemisphere EASE-Grid Weekly Snow Cover

File `snowcover.mon.mean.nc`

January

February

March

April

May

June

July

August

September

October

November

December

Other NetCDF Utilities

- Many other useful netCDF utilities developed by third parties are available:
  - NCAR Command Language (NCL)
    - https://www.unidata.ucar.edu/software/netcdf/software.html#NCL
  - NCO (NetCDF operators)
    - https://www.unidata.ucar.edu/software/netcdf/software.html#NCO
  - CDO (Climate Data Operators)
    - https://www.unidata.ucar.edu/software/netcdf/software.html#CDO

- For additional utility software, consult:
  - Unidata’s Software for Manipulating or Displaying NetCDF Data
    - http://www.unidata.ucar.edu/netcdf/software.html
Files for Practising

- **File simple_xy_nc4**
  - Write/Read the simple_xy file with some of the features of NetCDF-4.
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__nc4__wr_8c.html
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__xy__nc4__rd_8c.html

- **File simple_nc4**
  - Write/Read a file demonstrating some of the features of NetCDF-4.
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__nc4__wr_8c.html
  - https://www.unidata.ucar.edu/software/netcdf/docs/simple__nc4__rd_8c.html
Files for Practising

- **File sfc_pres_temp**
  - This is an example program which writes/reads surface pressure and temperatures.
  - [https://www.unidata.ucar.edu/software/netcdf/docs/sfc__pres__temp__wr_8c.html](https://www.unidata.ucar.edu/software/netcdf/docs/sfc__pres__temp__wr_8c.html)
  - [https://www.unidata.ucar.edu/software/netcdf/docs/sfc__pres__temp__rd_8c.html](https://www.unidata.ucar.edu/software/netcdf/docs/sfc__pres__temp__rd_8c.html)

- **File pres_temp_4D**
  - This is an example program which writes/reads 4D pressure and temperatures.
  - [https://www.unidata.ucar.edu/software/netcdf/docs/pres__temp__4D__wr_8c.html](https://www.unidata.ucar.edu/software/netcdf/docs/pres__temp__4D__wr_8c.html)
  - [https://www.unidata.ucar.edu/software/netcdf/docs/pres__temp__4D__rd_8c.html](https://www.unidata.ucar.edu/software/netcdf/docs/pres__temp__4D__rd_8c.html)
Files for Practising

Global Potential Vegetation Dataset

- File `vegtype_5min.nc` – NetCDF 5 min data
- File `vegtype_0.5.nc` – NetCDF data aggregated to a 0.5 deg resolution

Northern Hemisphere EASE-Grid Weekly Snow Cover

- File `snowcover.mon.mean.nc` – Monthly Mean
- File `snowcover.mon.ltm.nc` – Monthly Long Term Mean
- Files available at: https://psl.noaa.gov/data/gridded/data.snowcover.html
Summary of Actions

- Inspect the write and read files in C code.
- Compile and run the write/read C files.
- Inspect the output NetCDF file (.nc) using `ncdump`.
- Create a CDL file for the NetCDF file.
- Recreate the NetCDF file using `ncgen` and the CDL file.
- Recreate the C file using `ncgen` and the CDL file.
- Visualize the data in the NetCDF file with `ncview`.
- Change dimensions, variables, and attributes and rebuild the previous steps.
Appendix
The usual way of building NetCDF requires the HDF5, zlib, curl and m4 libraries.

Files for the libraries can be found in:

ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4

The following slides presents the steps for installing NetCDF in Ubuntu 18.04 and 20.04 for a user named *username*. Adapt the path to your own user.
Installing curl and m4

- `apt-get install libcurl4-openssl-dev`
- `apt-get install m4`
Installing zlib

  - Newest version to later use `ncview`
  - `wget https://sourceforge.net/projects/libpng/files/zlib/1.2.9/zlib-1.2.9.tar.gz`

- `tar -xvzf zlib-1.2.8.tar.gz`

- `cd zlib-1.2.8`

- `mkdir /home/username/local/

- `./configure --prefix=/home/username/local/

- make check install`
Starting with NetCDF

Installing HDF5

- `tar -xvzf hdf5-1.8.13.tar.gz`
- `cd hdf5-1.8.13`
- `./configure --with-zlib=/home/username/local/ --prefix=/home/username/local/`
- `make`
- `make check`
- `make install`
  - `make check install`
  - If not done separately, it might not work!
Installing NetCDF

- Check the latest version at https://www.unidata.ucar.edu/downloads/netcdf/
- `wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-c-4.7.4.tar.gz`
- `tar -xvzf netcdf-c-4.7.4.tar.gz`
- `cd netcdf-c-4.7.4`
- `CPPFLAGS=-I/home/username/local/include LDFLAGS=-L/home/username/local/lib .configure --prefix=/home/username/local`
- `make check install`
Finishing the Set Up

- Link the NetCDF library
  - `export LD_LIBRARY_PATH=/home/username/local/lib/`
  - `sudo ldconfig`
The ESiWACE1/2 projects have received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 675191 and No 823988

Disclaimer: This material reflects only the author’s view and the EU-Commission is not responsible for any use that may be made of the information it contains