GRIIDC Guide to Creating NetCDF files for CTD Data

NetCDF (Network Common Data Format) is a format for self-describing binary files, created and maintained by Unidata (http://www.unidata.ucar.edu/software/netcdf/) and is widely used in the earth sciences to store and exchange scientific data. Self-describing means there is sufficient information inside the file that the bytes inside can be interpreted correctly as numbers or characters and as single values, vectors, or multi-dimensional arrays without any external supplemental information. A nearly unlimited amount of supplemental information can be added to the file such as descriptions of the variables (name and units) and answers to all of the "who, what, when, where, and why" questions about the dataset as a whole. With sufficient additional information the file becomes stand-alone, meaning everything needed for someone to use the dataset is completely inside the file.

NetCDF files are structured in a way that computer programs can read discrete parts of the file. This means programs do not need to read the whole file from disk to gain access to a subset. Programs reading the data can run faster, and there is no performance penalty for including all related data in a single file, which makes storage and archiving easier.

NetCDF files employ a standards-based format where the supplemental information added to the file conforms to some specific community's standards on what information is needed and how it should be expressed. The standards GRIIDC follows for CTD data are found on the National Center for Environmental Information's NCEI's NetCDF Template V2.0 website. These are built-up from standards from four sources:

1) The NetCDF User's Guide (NUG)
   https://docs.unidata.ucar.edu/nug/current/

2) The NetCDF Climate and Forecast (CF) conventions
   http://cfconventions.org

3) The NetCDF Attribute Convention for Dataset Discovery (ACDD)
   http://wiki.esipfed.org/index.php/Attribute_Convention_for_Data_Discovery_1-3

4) The NCEI NetCDF Templates v2.0
   https://www.ncei.noaa.gov/data/oceans/ncei/formats/netcdf/v2.0/index.html

The template is a primary reference for this guide and is useful for those who do not have detailed knowledge of NetCDF. The following section on how to create a NetCDF file is intended to be brief and concise. Code templates in MATLAB, IDL, and Python will be provided in addition to a method that requires no programming, as it transforms a prepared text file directly into NetCDF.

Working with NetCDF files

The steps for creating standards-based NetCDF files are as follows:

1) Choose software tools.
2) Assemble the supplemental information and encode into “attribute=value” pairs.
3) Define the variables and “attribute=value” pairs and write them into the file.

NetCDF files are composed of two parts: variables and supplemental information. Variables are the data (like temperature), and supplemental information is everything else. Specify the data type (floating point number, integer, character, etc.) for each variable and, if the variable is an array, also specify its dimensions and the size of each dimension. Supplemental information is expressed as attribute=value pairs. Attributes are names, which come from the community standards; values are numbers or characters provided by the researcher based on knowledge about the data and the dataset. Each piece of supplemental information is assigned locally to a variable or globally to the dataset.

Step 1: Choose your software tools.

A plethora of free and commercial software is available to work with NetCDF files. This includes programming languages, command line utilities, and applications. Some of these tools can create NetCDF files. Others cannot and are only used to read NetCDF files and display their contents in various ways. Most programming languages can create NetCDF files. This guide recommends the programming approach. Guidance will be presented in generalized pseudocode and provided in actual codes in IDL, MATLAB, and Python. These examples should make it easy to port the examples into a specific language. For non-programmers, a demonstration follows on how to prepare a text file for transformation into a NetCDF file using the UNIDATA’s “ncgen” utility.

UNIDATA maintains a list of software that can manipulate NetCDF files in various ways and maintains several command line utilities that require no programming ability to use. UNIDATA’s frequently asked questions (FAQs) page for NetCDF is informative and comprehensive. The NetCDF utility “ncdump” is nearly indispensable for exploring NetCDF files. The quickest way to get ncdump and ncgen running is by way of binary distributions installed via one of the many package management systems available for a particular operating system.

UNIDATA’s List of software:
http://www.unidata.ucar.edu/software/netcdf/software.html

UNIDATA’s NetCDF Utilities:
https://www.unidata.ucar.edu/software/netcdf/workshops/most-recent/utilities/index.html

UNIDATA’s NetCDF FAQs:
https://docs.unidata.ucar.edu/netcdf-c/current/faq.html

Step 2: Assemble the supplemental information and encode into “attribute=value” pairs.

Supplemental information (e.g., ancillary information or metadata) provides additional information about the variables or the dataset itself. Supplemental information is inserted into NetCDF files as “attribute=value” pairs.

Here is an example of supplemental information for the variable temperature:

```
float swt(10)
swt:string:long_name     = “in situ temperature”
swt:string:standard_name = “sea_water_temperature”
swt:string:units         = “degrees C”
swt:string:comment       = “We used Dr. Spenser’s thermometer”
global:string:comment    = “Dataset not quality controlled.”
```

The first line defines a variable named “swt” as a single precision floating point array with ten elements. The name can be anything, provided it does not start with an underscore character. The next five lines are a series of “attribute=value” statements cast in the following pseudocode format:
scope : data-type : attribute name = value

The first part (swt or global) indicates the scope of the information – whether the attribute applies to the variable or to the whole dataset. A data type of “string” specifies the attribute value as a string of characters. Next is the name of the attribute and its value.

Attributes are names that come from the community’s standard list as well as those optionally added by the researcher. Values can be constrained (the entry comes from a standard list and must be spelled exactly as it appears in that list), unconstrained (the entry can be entered as the researcher likes), or data dependent (like the maximum longitude encountered in the dataset).

The attribute “long_name” is recommended by the CF convention, but its value “in situ temperature” does not need to come from a standard list. Guidance suggests this value should be set to something suitable for a plot title. The attribute “standard_name” is recommended and has the value “sea_water_temperature”. This value is constrained and must come from the CF standard name table. If a standard name does not exist for a variable, this attribute should be omitted. The attribute “units” is required and has the value “degrees C”. Values for “units” attributes are constrained and must appear in the udunits database to comply with the standard conventions. The swt attribute “comment” is optional, unconstrained, and can contain anything. The last line contains an attribute also named “comment”, but this attribute applies to the dataset as a whole as indicated by the term “global” in the scope position. Attribute names can only be used once in each variable and once in the global group.

Assembling the supplemental information is perhaps the most tedious part of building a NetCDF file. One reasonable way to complete this step is to cycle between the following three web sites as the attribute=value pairs are assembled:

1) The section "Standard Attributes and Guidance Table" on the NCEI NetCDF Templates 2.0 website lists the required and recommended attributes, their formats, a description of what is expected, and any constraints that need to be satisfied, along with additional guidance from NCEI.

2) The CF Standard Names list for filling in the attribute "standard_name" for variables. http://cfconventions.org/Data/cf-standard-names/30/build/cf-standard-name-table.html

3) A copy of the udunits database for filling in the attribute "units" for variables. https://github.com/Unidata/UDUNITS-2/blob/master/lib/udunits2-common.xml

Here are examples of some typical CTD variables:

Temperature:	standard_name = 'sea_water_temperature'	units = 'degrees_C'
Salinity:	standard_name = 'sea_water_practical_salinity'	units = 'PSU'
Conductivity:	standard_name = 'sea_water_electrical_conductivity'	units = 'S m-1'
Density:	standard_name = 'sea_water_density'	units = 'kg m-3'
Sigma-T:	standard_name = 'sea_water_sigma_t'	units = 'kg m-3'
Pressure:	standard_name = 'sea_water_pressure'	units = 'dbar'
Depth:	standard_name = 'depth'	units = 'm'
Latitude:	standard_name = 'latitude'	units = 'degrees_north'
Longitude:	standard_name = 'longitude'	units = 'degrees_east'

Note: Many oceanographers will say using PSU for practical salinity units is wrong; however, many users will understand the meaning. Feel free to use "g kg-1", "per mille", or "‰."

Step 3: Define the variables and "attribute=value" pairs and write them into the file.

NetCDF variables hold the CTD data. Recall that variables have the following properties: name, data-type, and dimensions. The name is unconstrained except it shouldn't start with the underscore character. Data-type indicates if it's a real number, integer, character, etc. Dimension specifies the shape and size of the variable. Variables can be scalar values, 1-D, 2-D, 3-D, or higher dimensioned arrays.

The following three code fragments written in IDL, Python, and MATLAB each define a variable named "sal" as a double precision vector with 100 elements:

; IDL

```
; Open a new file called 'testfile.nc'.  If it exists, clobber it (write over it).
; Define a dimension called 'z' and set to 100.
; Define the variable named 'sal' with dimension "z", and data-type=double
ncid = NCDF_CREATE('testfile.nc', /CLOBBER)
zid  = NCDF_DIMDEF(ncid, 'z', 100)
vid  = NCDF_VARDEF(ncid, 'Sal', zid, /DOUBLE)

; Same thing done in NetCDF4-Python.
; A double data-type in Python is indicated as an 8-byte floating point number 'f        8'
nc  = netCDF4.Dataset('testfile.nc', 'w')
z   = nc.createDimension('z',100)
sal = nc.createVariable('sal','f8',('z',))

; MATLAB code is almost identical to IDL code.
ncid  = netcdf.create('testfile.nc','CLOBBER')
zid   = netcdf.defDim(ncid,'z',100)
varid = netcdf.defvar(ncid,'sal','double',zid)

; MATLAB high-level functions do it all in one line.
nccreate('testfile.nc', 'sal','dimensions',{'z',100},'Datatype,'double')
```

Writing into a NetCDF file occurs in one of two distinct modes: define mode and data mode. Defining variables, dimensions, or attributes takes place in define mode. Loading data into variables is accomplished in data mode. Many programs will switch define mode automatically once a new NetCDF file is created. Issuing a command like the following for IDL and MATLAB will enter data mode. Note that program language may differ; Python and the MATLAB high-level functions seem to know what to do and shield any mode switching.

```
IDL          netcdf_control,ncid,/ENDDEF  ; switch from define mode to data mode
MATLAB       netdf.endDef(ncid)           ; switch from define mode to data mode
```

Note that the variable time for CTD data is expressed as seconds since "1970-01-01 00:00:00 0:00". Date-time information needs to be converted to this format. Use a double precision number to preserve the seconds information.

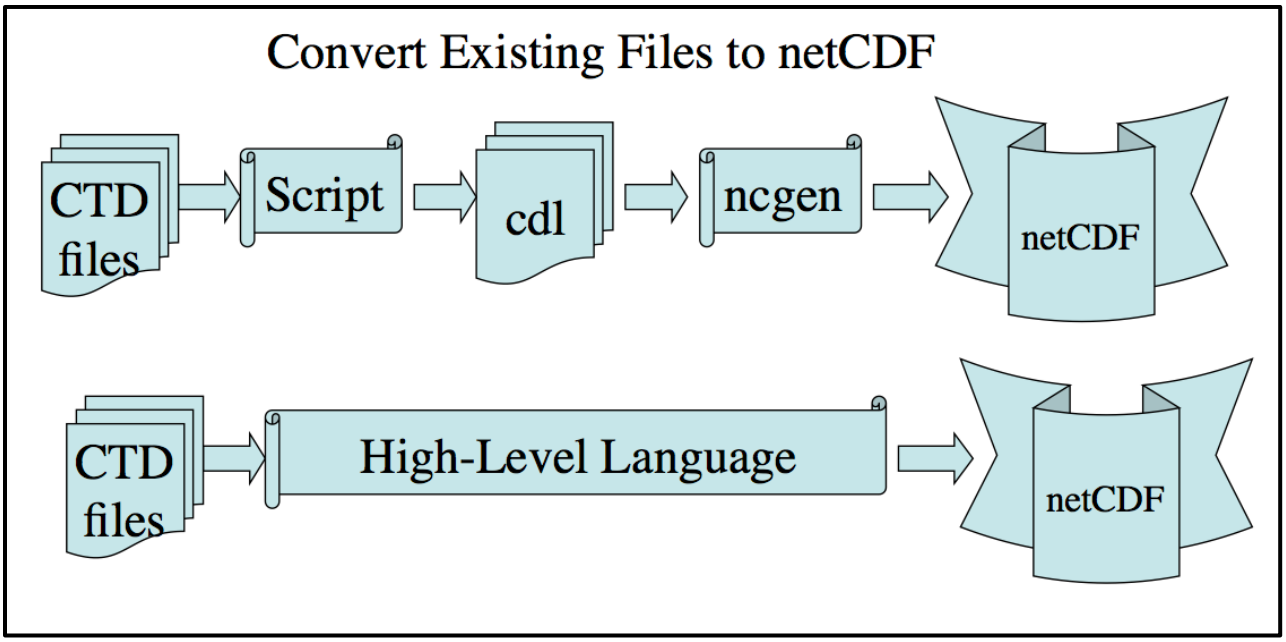
The typical workflow designed to process many datasets generally requires a main program to read the data from the original source and pass the data to a subroutine that does the following:

1) opens a new NetCDF file
2) loads the attributes with constrained and unconstrained values
3) computes the data dependent values for data dependent attributes

4) writes the attribute=value pairs into the NetCDF file
5) defines and writes the variables into the NetCDF file
6) closes the NetCDF file

The subroutine is modified for new projects by changing the global attribute values to suit each new project information.

The figure below illustrates two general approaches to generating NetCDF files. The second has been described above. The first uses UNIDATA's ncgen utility, which requires no programming language. In this case, the original CTD file is reformatted to an intermediate text file in common data language (CDL) form and then transformed using "ncgen" into a NetCDF file. The script is any tool that transforms the CTD data into a CDL form, perhaps in an automated way if processing a large number of files.



Here is a very simple example of a CDL file containing a time series of daily river discharge data. It begins with the word "netcdf" followed by a filename "riverdischarge" and an opening bracket. There are a few section identifiers "dimensions", "variables", and "data". Each section identifier is followed by attributes and values. For example, "flow" is a floating point number that has as many values as there are year values. Flow has several attributes: a long name, a missing value specification, and units. There is no limit on the amount of optional metadata (messages) that can be included. The variables are loaded with their values and the block ended with a closing bracket. If all the punctuation is correct, the UNIDATA utility ncgen would transform this text file into a NetCDF file named "riverdischage.nc".

```
netcdf riverdischarge {
dimensions:
year = unlimited;
```

```
variables:
float flow(year);
flow:long_name      =  "Average daily river discharge";
flow:missing_value = -9.9900e+02;
flow:units              =  "cubic meters per day";
integer year(year);
year:long_name      =  "Year";
integer month(year);
month:long_name   =  "Month";
integer day(year);
day:long_name       =  "Day";
:usgs_gauge         =  02479310;
:usgs_location      =  "PASCAGOULA RIVER AT GRAHAM FERRY, MS";
:message1           =  "RIVER DISCHARGE VOLUME (CUBIC METERS PER DAY)";
:message2           =  "DATES ENCOMPASSED:  10/01/1993 to 04/05/2001";
:message3           =  "CONTACT: MISSISSIPPI USGS, D. P Buddy.";
:message4           =  "SOURCE: http://water.usgs.gov/ms/nwis/sw";
:message5           =  "NOTE: DATA AFTER 30-SEPT-1999 ARE PROVISIONAL";
:message6           =  "NOTE: MISSING VALUES FLAGGED WITH  -9.9900e+02";
data:
year     = 1993,1993,1993,1993,1993,1993,1993,1993,1993,1993;
month = 10,10,10,10,10,10,10,10,10,10;
day      = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10;
flow     = 5.2357e+06,  5.1867e+06,  5.0155e+06,  5.0155e+06,  5.1378e+06,  5.0155e+06,  4.9421e+06,
4.8687e+06,  4.7464e+06,  4.7219e+06;}
```

The only difference between this example and a standards-based NetCDF file conforming to NCEI preferences is it required more metadata and the use of standard names (flow is not a standard name). NCEI has prepared a CDL template for CTD data:

https://www.ncei.noaa.gov/data/oceans/ncei/formats/netcdf/v2.0/profileOrthogonal.cdl

All that is missing from this template is the section where the data are defined. This template can be edited to include specific information, adding a specific set of data at the bottom; ncgen can then create the NetCDF file.

One final note, the NCEI orthogonal profile template referenced above is set up for multiple CTD profiles. Because the guidance illustrated only one cast per NetCDF file, the dimensionality of the template code was reduced by replacing “float lat(profile)” with “float lat” and “geophysical_variable_1(profile, z)” forms with “geophysical_variable_1(z)” forms. To store multiple CTD casts in the same NetCDF file in the above code examples, the term “profile” would be the number of CTD casts stored in the NetCDF file.