Harper Simmons

NETCDF OPERATORS

Workshop Module 6

INSIDE THIS MODULE:

INTRODUCTION AND HISTORY

NCO is a suite of programs known as operators created by Charlie Zender. Each operator is a stand-alone, command line program that is executed at the UNIX shell-level. The operators take netCDF file(s) as input, perform an operation (e. g. averaging), and produce a netCDF file as output. The operators are primarily designed to aid manipulation and analysis of data. NCL or Fortran could be used to accomplish much of the same processing, but the NCO operators have been designed to perform specific functions in an efficient manner with simple interfaces. Although the operators were originally designed to process climate data, they have evolved into a general netCDF tool. NCO versions exist for many operating systems including GNU/Linux, SunOS 4.1x, Solaris 2.x, IRIX 5.x, and 6.x, and Windows. The primary requirement to run NCO is the correct installation of the netCDF libraries.

http://nco.sourceforge.net/

APPENDING VS. CONCATENATION

The terms appending and concatenating are similar terms and are often confused with respect to NCO which treat them very differently. Appending or merging is the adding of variables from one file to another. For example, one file contains the variables T, U, V and another file contains the variables MAG and DIR which were derived from U and V, so that they exist for the same points in both time and space. Appending would result in the combination of these two files into a new file that contains all 5 variables. **One should use the ncks operator to append variables.** Concatenation on the other hand occurs when NCO combines variables along their record dimension. This would occur if there were two files each containing one-year of temperature data. A concatenation operation would produce a new file with just one variable, T that spanned two years. In this instance, the variables shared a record dimension (time). **ncrat** specifically designed to concatenate variables along their record dimension (think "r" for record). Now, if the variables in the files were averages, it is possible that they would no longer have a time dimension. The NCO **ncecat** would create a single variable along a new record dimension, by default called "record". In this case, think "e" for ensemble.

SWITCHS COMMON TO VARIOUS OPERATORS

Switchs are command line options that allow the user to modify an operators functionality. Many of the switches are common to more than one operator, so they are presented separately here even though the operators themselves have not been individually discussed.

"-n" and "-p"

These switches assist the user is specifying which files to operate on. There are four different ways of specifying input files to NCO, explicitly typing each (example 1), using UNIX shell wildcards, and using the NCO "-n" and "-p" switches. The "-p" switch (example 2) is used to specify the directory where all the input files reside, so that it only has to be input once. NCO then prepends input-path (e.g.,"/data/usrname/model") to all input-files (but not to output-file). Note input-path need not end with "/"; the "/" is automatically generated if necessary.

Example 1: Explicit typing of the files. Operator initiated in the directory where the data is located.

ncra 001.nc 002.nc output.nc

Example 2: Use of "-p". Operator is initiated in a non-local directory. The "-p" points to the data. The output file is created in the directory from which the operator is initiated.

ncra -p /data/usr/ 001.nc 002.nc output.nc

The "-n" switch (example 3) allows the user to concisely describe an entire set of filenames. This option is only available with the multi-file operators: ncra, ncrcat, ncea, and ncecat. Multi-file operators process an arbitrary number of input-files.

Example 3: Use of "-n". The syntax tells NCO to construct five (5) filenames indentical to "85.nc" except that the final two (2) digits are a numeric suffix to be incremented by one (1) for each successive file. The `.nc' and `.cdf' suffixes, if present, are not counted toward the two digit numeric suffix.

ncra -n 5,2,1 85.nc out.nc

"-R"

NCO automatically deletes files retrieved from remote locations after they have been processed in order to conserve local file system space. Many NCO operators were constructed to work with numerous large files. Retrieval of multiple files from remote locations is done serially. Each file is retrieved, processed, then deleted before the cycle repeats. In cases where it is useful to keep the remotely-retrieved files on the local file system after processing, the automatic removal feature may be disabled by specifying "-R" on the command line.

"- v" and "-x"

Variable selection is implemented with the "-v var[.,.,.]" (example 4) and "-x" (example 5) switches. A list of variables to extract is specified following the "-v" option, e.g., "-v T,U,V". Not using the "-v" option is equivalent to specifying all variables. The "-x" option causes the list of variables specified with "-v" to be excluded rather than extracted. Thus "-x" saves typing when you only want to extract fewer than half of the variables in a file.

Example 4: Extract a list of variables from one file and place in another:

ncks –v T,U,V in.nc out.nc

Note: out.nc will contain not only the variables T,U, and V, but also all the coordinate variables associated with those variables. (See switches "-c" and "-C" on page 3)

Example 5: Extract all the variables from one file to another EXCEPT for the variables listed:

ncks –x –v CHI,PSI in.nc out.nc

"-c" and "-C"

By default, coordinates variables associated with any variable appearing in the input-file will also appear in the output-file, even if they are not explicitly specified, e.g., with the "-v" switch. Thus variables with a latitude coordinate lat always carry the values of lat with them into the output-file. This is a useful graphical feature since many of NCL's high level interfaces look for these variables. This feature can be disabled, however, with "-C".

The "-c" option, on the other hand, is a shorthand way of automatically specifying that all coordinate variables in the input-files should appear in the output-file.

"-F"

By default, NCO uses C-style (0-based) indices for all I/O. The "-F" switch tells NCO to switch to reading and writing with Fortran index conventions. In Fortran, indices begin counting from 1 (rather than 0), and dimensions are ordered from fastest varying to slowest varying.

"-r"

All operators can be told to print their internal version number and copyright notice and then quit with the "-r" switch. The internal version number varies between operators, and indicates the most recent change to a particular operator's source code. This is useful in making sure you are working with the most recent operators.

"-d"

The "-d" option will extract a subset of data (called a hyperslab). The coordinates of a hyperslab are specified with the dimension named followed by the minimum and maximum values to extract.

Example 6: Extract the first 6 time records of in.nc and place them in out.nc.
ncks –d time,0,5 in.nc out.nc

A half-open range is specified by omitting either the min or max parameter but including the separating comma. The unspecified limit is interpreted as the maximum or minimum value in the unspecified direction.

Example 7: Example 6 can be rewritten as:
ncks –d time,,5 in.nc out.nc

Extract the fifth through last record:
ncks –d time,5,, in.nc out.nc

If values of a coordinate-variable are used to specify a range, then the coordinate variable must be monotonic (values either increasing or decreasing). It is possible that a previous extraction resulted in a non-monotonic variable (Example 8). Ranges are determined by seeking the first coordinate value to occur in the closed range [min,max] and including all subsequent values until one falls outside the range. The "-d" option may be specified more than once. **ncks** allows for a stride option as well which follows the max indicator. **ncra** and **ncrcat** also allow a stride argument, but only for the record dimension.

Example 8: Coordinate values should be specified using real notation, whereas dimension indices are specified using integer notation.
ncks -d lon,340.,50. -d lat,10.,35. in.nc out.nc

The results of the above operation result in a lon dimension that is not monotonic (e.g. 340,350, 10,20 etc). This will pose a problem for NCL which assumes monotonic coordinate variables.

"-A" and "-O"

If the output-file specified for a command is a pre-existing file, then the operator will prompt the user whether to overwrite (erase) the existing output-file, attempt to append to it, or abort the operation. NCO also implements two ways to override its own safety features, the "-O" and "-A" switches. Specifying "-O" tells the operator to overwrite any existing output-file without prompting the user interactively. Specifying "-A" tells the operator to attempt to append to any existing output-file without prompting the user interactively.

"-h"

All operators automatically append a history global attribute to any file they modify or create. The history attribute consists of a timestamp and the full string of the invocation command to the operator, e.g., "Mon May 26 20:10:24 1997: ncks in.nc foo.nc". The full contents of an existing history attribute are copied from the first input-file to the output-file. The time stamps appear in reverse chronological order, with the most recent timestamp appearing first in the history attribute. Since NCO and many other netCDF operators adhere to the history convention, the entire data processing path of a given netCDF file may often be deduced from examination of its history attribute. To avoid information overkill, all operators have an optional switch ("-h") to override automatically appending the history attribute.

MISSING VALUES

Missing data refers to data points that are missing, invalid, or for any reason not intended to be arithmetically processed in the same fashion as valid data. Missing data is identified by the variable's **missing_value** attribute. If this attributes exists, then any elements of the variable which are numerically equal to the missing_value are treated as missing data. If the "type" of the attribute missing_value is not the same as the "type" of the variable to which it applies, NCO converts the value. When an NCO arithmetic operator is processing a variable with a missing_value attribute, it compares each value of var to missing_value before performing an operation. This inflicts a performance penalty on the operator even when none of the data is missing.

Do not assign a missing_value to any variable that does not have missing data in order to avoid a performance hit.

Note: NCL uses the attribute _FillValue as its designation of missing data. Depending upon the data processing requirements, it may be necessary to assign both values to a variable.

ACKNOWLEDGMENTS:
Most of the text in this module has been taken directly from Charlie Zender's *NCOs User's Guide.* The web address for this Guide is located on the front page of this module. Everyone who process netCDF data owes a debt of gratitude to Charlie for creating these useful operators.

THE OPERATORS

NCATTED ATTRIBUTE EDITOR

ncatted edits attributes in a netCDF file. It can append, create, delete, modify, and overwrite attributes. It allows each editing operation to be applied to each variable in a file.

Example 1: Append the character string "test of ncatted" to the global history attribute.

ncatted –a history,global,a,c," test ncatted" in.nc

att_nm=history
var_nm=global
mode=append
att_type=character
att_val="test of ncatted"

Note the space between the " and the "t" in the string. **ncatted** appends directly to the last character of the attribute.

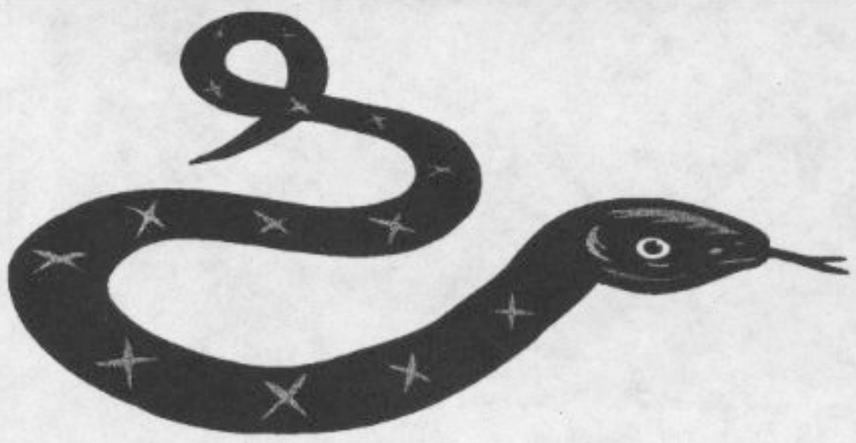
Note: the history attribute of in.nc will look like the following:

Fri Oct 1 13:58:11 1999: ncatted –a history, global,a,c,test of ncatted in.nc" "Data created in NCOM on 01/26/99 08:56:28 test of ncatted.

In essence the phrase has been added to the history attribute because NCO always repeats its commands in this attribute. To suppress this, add a "-h" option.

ncatted –h –a history,global,a,c," test" in.nc

Data created in NCOM on 01/26/99 08:56:28 test.



ncatted [-a **att_dsc**] [-h] [-l path] [-O] [-p path] [-R] [-r] input-file [output-file]

Where: ***(Note: these are ORDER dependent)***
att_dsc = att_nm,var_nm,mode,att_type,att_val

att_nm: The name of the attribute to edit e.g. ***units*** or ***long_name***

var_nm: The name of the variable to edit. A blank implies all variables. "global" = global attribute.

mode: Is a single character abbreviation standing for one of the following:
a = append
c = create
d = delete
m = modify
o = overwrite

att_type: Is a single character abbreviation standing for one of six data types:
f =float
d = double
l = long
s = short
c = char
b = byte

att_val: Is what you want the value changed to. It may be a single value or a 1D array. When specifying an array, or a character string, enclose the values in quotes.

Example 2: Delete all existing units attributes.

ncatted –a units,,d,, in.nc

att_nm=units
var_nm=blank so that all variables are effected
mode=delete
att_type=blank because not needed
att_val=blank because not needed

Example 3: Change the value of the long_name attribute for the variable T from temperature to TEMP.

ncatted –a long_name,T,o,c,TEMP in.nc

att_nm=long_name
var_nm=T
mode=overwrite
att_type=character
att_val="TEMP"

NCEA AND NCRA

NCEA
ENSEMBLE AVERAGER

ncea [-A] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-n loop] [-O] [-p path] [-R] [-r] [-x] [-v...] input_file(s) output_file

ncea performs gridpoint averages of variables across an arbitrary number (ensemble) of input files. While **ncra** only performs averages over the record dimension (e.g. time), **ncea** averages entire files and weights each file evenly. Each variable in the output_file will be the same size as the same variable in any of the input files. **ncea** does not average the coordinate variables.

Example 1: Average the Jan model files from three different model runs.

ncea j1.nc j2.nc j3.nc ave.nc

Example 2: Calculate the same average, but only for the variables T, and U. Also, turn off the automatic history attribute.

ncea –h –v T,U jan_1 jan_2 jan_3 ave.nc

Example 3: Calculate the example 1 average but only over a geographic subregion, and put in a switch to automatically overwrite ave.nc if it exists without prompting the user.

ncea –O –d lat,-20,20 –d lon,180,360 jan_1 jan_2 jan_3 ave.nc

7/22/08
Final file has same # time as orgmal files.

NCRA
RECORD AVERAGER

ncea [-A] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-n loop] [-O] [-p path] [-R] [-r] [-x] [-v...] input_file(s) output_file

ncra averages record variables across an arbitrary number of input files. The record dimension is retained as a degenerate (size 1) dimension in the output variables. This dimension must be removed for **ncdiff** to work when using this mean file to create anomalies (see page 8). The record coordinate should be monotonic. See page 5 for an instance when monotonicity is lost. **ncra** weights each record (e.g. time slice) in the input_files equally.

Example 1: Average the three months of model data to create a seasonal file.

ncra 12.nc 01.nc 02.nc DJF.nc

Example 3: Create a csh script that reads in one year of data and averages it. The files have the appearance of 0001-01.nc.

```
#! /bin/csh –f

set yr = "0001"
set mo_strt = "01"
set num_mons = 12

@ mo = $mo_strt
@ mo_end = $mo_strt + num_mons –1

while ( $mo <= $mo_end )
  set fn = `printf "%04d" ${yr}`-`printf "02d" ${$mo}`
  msread $fn /JOEBLOW/$fn
  @ mo++
end

ls ${yr}*.nc file.txt
set files = `cat file.txt`
ncra –O $files ${yr}.nc
```

NCECAT, NCRCAT, AND NCFLINT

NCECAT
ENSEMBLE CONCATINATOR

ncecat [-A] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-n loop] [-O] [-p path] [-R] [-r] [-x] [-v...] input_file(s) output_file

ncecat concatenates an arbitrary number of input files into a single output file. Input files are glued together by creating a **record** dimension in the out output file. Input files must be the same size. Each input file is stored consecutively as a single record in the output file. In the preparation of this module, it was found that **ncecat** also concatenated variables such as hybm. This variable is the same in every file, so it duplicates its values. The user will have to be aware of this if using NCL to extract hybm and similar variables because the dimension of this 1D variable will become a 2D variable.

Example 1: concatenate two files that contain derived variables.

ncecat derive_1.nc derive_2.nc tot.nc

NCRCAT
RECORD CONCATINATOR

ncrcat [-A] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-n loop] [-O] [-p path] [-R] [-r] [-x] [-v...] input_file output_file

ncrcat concatenates record variables across an arbitrary number of input files. The final record dimension is by default the sum of the lengths of the record dimensions in the input files. Input files may vary in size, but each must have an UNLIMITED record dimension. If the files have a record dimension, but it is NOT UNLIMITED, then **ncrcat** will fail, and it will be necessary to use **ncecat**. This is one reason why in the creation of netCDF files it is always important to ensure the record dimension is UNLIMITED even if there is no anticipation of joining files.

Example 1: Concatenate two files that contain one month of data each. 01.nc: (time:1-12) 02.nc: (time:13:24)

ncrcat 01.nc 02.nc 01-02.nc

NCFLINT
LINEAR INTERPOLATOR

ncflint [-A] [-C] [-c] [-d ...] [-F] [-h] [-i var,val3] [-l path] [-O] [-p path] [-R] [-r] [-v ...] [-w wgt1 [,wgt2]] [-x] *input_file* [*output_file*]

ncflint creates an output file that is a linear combination of the input files. This combination can be a normalized weighted average, or an interpolation of the input files specified with a "-l" option. In the first method, the user specifies weights *"-w wgt1* [*wgt2*]". If *wgt2* is not specified, it is automatically computed as (*wgt2* = 1 - *wgt1*). Weights larger than 1 are authorized. **ncflint** then takes the input values, multiples them by their respective weights, and then adds the results together. In the second method, the user does not know the respective weights, but it does now the resultant target (e.g. where you want to interpolate to). Given this value *[-i var,val3]*, **nclfint** will determine the values of *wgt1* and *wgt2* and apply them to all variables selected. The variable var must represent a single scalar value so that if it has dimensions, they must be degenerate. If neither the "-w" or "-i" option are specified, then **ncflint** defaults to weighting each file evenly.

Example 1: Interpolate in time to a value between two files.

ncflint –i time,2 1.nc 3.nc

In this simple case, we would expect the weights to be even since the target falls exactly between the two input files. In most interpolations, however, this may not be the case, which makes the "-l" option so useful.

Example 2: Create a wind forcing data set through the linear combination of a coastal timeseries and an offshore bouy. Adjust the weights of the two locations that most accurately reproduce the coastal wind driven flow.

nclfint –w .3 coastal.nc bouy.nc

NCDIFF AND NCRENAME

NCDIFF
DIFFERENCER

ncdiff [-A] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-O] [-p path] [-R] [-r] [-x] [-v...] file_1 file_2 file_3

ncdiff subtracts variables in file_2 from the corresponding variables in file_1 and places the results in file_3. If the variables are not of the same dimension, then the variables in file_2 are expanded to conform to the dimension size of the corresponding variable in file_1. This allows the operator to calculate anomalies from a mean. The dimensions that variable_1 from file_1 and variable_2 from file_2 have in common must be exactly the same size. This means that if the mean file contains a time dimension of dimension 1, it will cause **ncdiff** to fail, because it does not match the dimension of variable_1 from file_1. It is therefore necessary to remove the time dimension from the mean file to avoid this problem (example 2). **ncdiff** will never difference coordinate variables.

Example 1: Difference two files of equal length

ncdiff 001.nc 002.nc diff.nc

Example 2: Calculate an temparature anomaly by first calculating a mean file

ncra 001.nc 002.nc ... 012.nc year.nc

(note: see page 6 For more details on ncra)

ncra produces a time dimension which must be removed:

ncwa –O –a time year.nc year.nc

This command removes the time dimension of size 1 left by ncra

Now we can create the anomaly file:

ncdiff –v T 001.nc year.nc anom.nc

NCRENAME
RENAMER

ncrename [-a old_name, new_name] [-a ...] [-a old_name, new_name] [-d ...] [-h] [-l path] [-O] [-p path] [-R] [-r] [-v old_name, new_name] [-v...] in_file out_file

ncrename renames dimensions, variables, and attributes in a netCDF file. Each object that has a name in the list of old names is renamed using the corresponding name in the list of new names. All the new names must be unique. Every old name must exist unless it is preceded by a ".". The validity of the old_names is not verified, so if an old_name does not exist, and it is not specified with a ".", **ncrename** will abort. If only an in_file is specified, the file will be overwritten without prompting the user.

attribute renaming: "-a old_name, new_name"

dimension renaming: "d old_name, new_name"

variable renaming: "-v old_name, new_name"

Example 1: rename the variable "p" to "press", and the variable "t" to "temp". In this case, "p" must exit in in_file, but because of the "." It is optional for "t".

ncrename –v p,press –v .t,temp in_file.nc

Example 2: Create a new file with the attribute long_name of every variable changed to title.

ncrename –a long_name,title in.nc out.nc

(note: which other NCO utility can rename an attribute? Answer=ncatted)

NCKS

NCKS
KITCHEN SINK

ncks extracts a subset of the data from the *input_file* and either prints it as ASCII text to the screen, or writes it to *output_file*, or both. When **ncks** prints the data to the screen, it does so in a tabular format that is easier to read than ***ncdump***. When **ncks** extracts a variable from the *input_file*, and pastes it to an existing netCDF file, then the global attributes of the existing file are overwritten. Since the history of a file is contained in the *global@history* attribute, this feature can potentially remove important information. Additionally, if the user attempts to append a variable from the *input_file* that already exists in the *out_put* file, it will be overwritten. If the record dimensions of the two files are different, **ncks** will create a new record dimension in the *output_file* that is the largest of the two dimensions and use fill values as necessary.

ncks [-A] [-C] [-c] [-d ...] [-F] [-H] [-h] [-l path] [-M] [-m] [-O] [-p path] [-R] [-r] [-s format] [-u] [-v ...] [-x]
input_file [*output_file*]

"-M","-m","-H", "-F", "-s", and "-u" are switches unique to **ncks** that control the formatted appearance of the data extracted.

"-H": print data to screen.

"-M": print the global metadata (summary info and global attributes) to the screen.

"-m": print variable metadata to the screen.

" -s format": accepts C printf() formats.

"-u": print the variable name along the units attribute if it exists.

EXTRACTIONS

Example 1: Extract variables TS and V from the input file and place them in a new output file. If the output file exists, overwrite it without prompting the user.

ncks –O –v TS,V in.nc out.nc

Example 2: Extract a subset of the variable PS in both time and space and append it to the above output file.

ncks –A –d time,5 –d lat,–20.,20. –d lon,,180. –v PS

In this instance, the fifth index of the time array is selected. Since it is an index and not an actual coordinate value, it is in integer form. The latitudinal subset goes from 20S to 20N, and the longitudinal subset includes everything greater than 180.

PRINTING TO THE SCREEN

Example 3 : print TS in the input file using Fortran indexing conventions:

ncks –H –F –v TS in.nc

e.g. lon(1)=0 lat(1)=-87.86 time(1)=1613 TS(1)= 246.193

Example 4: Now add the variables units and change to C indexing conventions:

ncks –H –u –v TS in.nc
e.g. time[0]=1613 lat[0]=-87.86 lon[0]=0 TS[0]=246.193 K

(note the that the order the variables are printed is reversed. The fastest varying dimension becomes the slowest varying. This change also occurs when switching conventions in NCL.

NCWA

NCWA
WEIGHTED AVERAGER

ncwa averages variables in a single file over arbitrary dimensions which is unlike **ncea** (page 6) which averages over all files, and **ncra** (page 6) which averages only over the record dimension. There are options to specify weights, masks, and normalizations. The mask and weights are broadcast to conform to the dimension of the variables being averaged. The rank of the variable is reduced by the number of dimensions which they are averaged over.

Example 1: Calculate an average over all ocean values

ncwa –m ORO –M .5 –o lt –a lat,lon in.nc out.nc

Here the mask is ORO, the value to compare to is 0.5, and the comparison operation is lt (less than).

Example 2: Calculate a zonal average

ncwa –a lon in.nc out.nc

Example 3: Calculate a meridional average weighted by the gaussian weights.

ncwa –w gw –a lat in.nc out.nc

ncwa [-A] [-a dim] [-C] [-c] [-d ...] [-F] [-h] [-l path] [-m val] [-O] [-n] [-N] [-p path] [-R] [-r] [-u] [-W] [-x] *input_file* [*output_file*]

Masking condition:
-m variable
-M value
-o condition (eq,ne,gt,lt,ge, le)

Normalization:
-N off
-n by tally only
-W by weight but not tally

This document was written to provide students of the NCAR Data Processing and Visualization Workshop with a reference manual on the netCDF operators. Other modules include discussions on the netCDF data model (module 1), NCL Basics (module 2), NCL File IO (module 3), Data Processing with NCL (module 4), incorporation Fortran into NCL (module 5), and the Component Model Processing Suite (CMPS) (module 7). All modules are downloadable from the internet at:

http://www.cgd.ucar.edu/csm/support/Document/manual.shtml

Community Climate System Model Support Network

National Center for Atmospheric Research

CGD/CAS
1850 Table Mesa Drive
Boulder, CO 80305

Phone: 303-497-1720
Fax: 303-497-1333
Email: murphys@ucar.edu

Community Climate System Model
Support Network
National Center for Atmospheric Research