# NetCDF Operators and Utilities

**Russ Rew**
**Steve Emmerson**

This draft document describes an initial set of netCDF operator and utility programs.

NetCDF **operators** read one or more netCDF input files and write a netCDF output file. NetCDF **utilities** read one or more netCDF files and produce some other kind of output (*e.g.* a CDL file or graphics), or read some other kind of input (*e.g.* a CDL file) and write netCDF files. The notion of a netCDF operator is derived from the analogous concept in Raymond's (1988) Candis system, and the specific operators proposed here are similar in function to some of the operators specified in Dave Raymond's *Proposal for NetCDF Algebra*, 1989.

## 1.0 General Conventions for Programs

Each program will accept, as appropriate, positional parameters naming one or more input netCDF files and one output netCDF file.

If an output file does not already exist, a netCDF operator will create it. If the output file exists, it will be overwritten only if the operator completes successfully. Similarly, if the output file is the same as one of the input files, the input file will be overwritten only on successful completion of the operator. Although this behaves as if the operation is being done "in place," the output is typically written to a temporary file, which is then renamed to the specified output file name on completion. Specifying an output file to be the same as one of the input files is a potentially unsafe operation, and should be avoided unless the input file can be easily recreated.

The netCDF programs use the concept of **coordinate variables**. A coordinate variable is a one-dimensional variable having the same name as its dimension. The purpose of coordinate variables is to store coordinate values for each grid point of a dimension. Thus, coordinate variables indicate the grid over which data are defined. Some netCDF programs assume that a dimension has an associated coordinate variable with monotonic (but not necessarily increasing or evenly-spaced) values as a function of dimension index, so that values may be interpolated or compared. Such programs will not work well with coordinate systems that have internal discontinuities, e.g. longitudes across the International Date Line. The `ncsort` operator can be used to make non-monotonic coordinates monotonic.

As a consequence of using coordinate variables, netCDF programs use generalized **dimensional coordinates**. A dimensional coordinate can be specified as either an integer dimension index, or, if an associated coordinate variable exists, as the corresponding floating-point value of the coordinate variable. If a dimension has coordinate variables in all input files, then dimensional coordinates may be given to an program as either integer indices or as floating-point values; otherwise, only integer indices must be used.

Variables, dimensions, and attributes are specified on command lines by name. Programs that require dimension values may provide ways to specify these values either in terms of the corresponding coordinate variable values or in terms of zero-based indices. In such cases, a convention used by the netCDF programs is that coordinate values should be specified using real notation with a decimal point required in the value, whereas dimension indices are specified using integer notation without a decimal point. Note that this convention is only to differentiate coordinate values from dimension indices, and is independent of the actual type of netCDF coordinate variables, if any.

In general, the global attribute `history` is used to keep a record of the operations that have been performed to generate a file. The `history` attribute will be constructed from input histories and the program invocation line. If there are multiple input files, their `history` attributes are merged on the output before appending the invocation of the command that merges them. This default behavior can be overridden by specifying `-h` on the command line, in which case no `history` attribute is created for the output file.

Other variable and global attributes are preserved in the output file when practical and meaningful. In the case of multiple input files with global attributes having the same name, only a single instance of the attribute is output if the input attribute values are identical; otherwise multiple instances of the global attributes suffixed with `_0`, `_1`, and so on are output, where the appended numerals specify the position of the input file in the invocation line, as recorded in the invocation string in the `history` attribute.

References to fill values are to variable-specific values of the `_FillValue` attribute, when such exists, otherwise to the type-specific fill values defined in the file `netcdf.h`. Fill values and `valid_range` attributes (when they exist in the netCDF) are properly propagated.

The `units` attribute is used by programs in deriving units for output variables and determining commensurability for operations on input variables.

See the Deferred section at the end of this document for a description of other planned conventions that may eventually be supported by netCDF programs.

# 2.0 Selectors

These operators can be used to select a subset of data from a netCDF file.

## 2.1 ncextr

**ncextr** **[-v** *var1* **[,...]]**

```
[-g att1[,...]]
[-x gv]
[-e var,vatt1[,...]] [...]
[-i var,vatt1[,...]] [...]
[-chs]
infile outfile
```

Extracts a subset of variables and global attributes from the input file and puts them in the output file. The options `-v` and `-g` specify, respectively, the set of variables and global-attributes to be included in or excluded from the output. Not specifying these options is equivalent to specifying all associated items in the input file. The `-x` option indicates that the specified set of variables (`v`) or global-attributes (`g`) is to be excluded from (rather than included in) the output: all items not in the respective set will be copied to the output. The `-e` option specifies that the attributes *vatt1,etc.* of variable *var* should not be copied. This option may be specified more than once.The `-i` option specifies that the attributes *vatt1,etc.* of variable *var* should be copied if the variable itself will appear in the output. This option may be specified more than once.

If the `-c` option is specified, coordinate variables associated with the dimensions of the extracted variables are also included in the output (even if such a variable is explicitly excluded).

The `-h` option specifies that the output netCDF is not to contain a history global-attribute.

The `-s` option causes the program to run silently: no warning messages are emitted.

Dimensions that are not used by any of the extracted variables do not appear in the output file; thus invoking `ncextract` with no variable or attribute lists will copy the input to the output except that unneeded dimensions will be deleted.

This operator is the inverse of `ncmerge`.

## 2.2 nccut

```
nccut -d name,[min][,[max]] [-d ...] infile outfile
```

Selects a subspace of the input file where the ranges on dimensions are given by the associated *min* and *max* values. A half-open range is specified by omitting either the *min* or *max* parameter but including the separating comma. The limit is then the maximum or minimum possible in the unspecified direction. A cross-section at a specific coordinate is extracted by specifying only the *min* limit and omitting a trailing comma. Dimensions not mentioned are passed with no reduction in range. The dimensionality of variables is not reduced (in the case of a cross-section, the size of the constant dimension will be one).

If values of a coordinate-variable are used to specify a range or cross-section, then the coordinate variable must be monotonic (values either increasing or decreasing). In this case, command-line values need not exactly match coordinate values for the specified dimension. 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 coordinate value for a cross-section is the coordinate-variable value closest to the specified value and must lie within the range of coordinate-variable values.

This operator is the inverse of `ncpaste`.

## 2.3 ncrdim

**`ncrdim -d`** *name*`,`**`[`***min***`][,[`***max***`]] [-d`** *...***`]`** *infile outfile*

Reduces the dimensionality of variables that use the specified dimensions by averaging the variables over dimension ranges or by taking cross-sections at single dimensional values. Such dimensions do not appear in the output. The syntax of and constraints on dimension-range and cross-section specifications are the same as for the `nccode` operator.

Missing data is ignored.

If a dimension has a coordinate-variable, then those values are used to weight the average using Simpson's (*i.e.* the trapezoidal) rule.

Affected variables will each have attributes attached to them specifying the dimensional reduction. Each reduction dimension will cause a identically-named attribute to be created whose value will be equal to the mean coordinate value for that reduction interval.

This operator is the inverse of `ncappend`.

## 2.4 ncthin

**`ncthin -d`** *name*`,`*step***`[,`***start***`] [-d`** *...***`]`** *infile outfile*

Thins the data by selecting every *step* grid point of dimension *name* beginning at point *start*, *etc*. If *start* is specified using a coordinate variable, then the coordinate variable must be monotonic. Only one specification per dimension is allowed. *start* defaults to the first point. Thus, `-var x,2,10.0 -dim y,3,1` keeps every second point of dimension `x` (starting with the first point whose `x` coordinate variable is greater than or equal to 10.0) and every third point of dimension `y` (starting with the second point). Dimensions which are not mentioned, or which are specified with a thinning factor of 1 and start at the first point, are not affected.

## 2.5 ncisocut

**`ncisocut -dim`** *dimname* **`-v`** *test_var* **`-x`** *test_val1***`[,`***...***`]`**
**`-t`** *file*
**`[-max|-min]`** *infile outfile*

Inverts the dependent/independent relationship between a variable and one of its dimensions by linearly interpolating an independent dimension to coordinates of a dependent variable. If the independent dimension has a coordinate variable, then that variable must be strictly monotonic. Inversion parameters can be specified using one of two methods: the `-d`, `-v`, and `-x` option, or the `-t` option, but not both.

In the first method, *dimname* is the independent dimension which will be interpolated to the *test_val1[,...]* coordinates of dependent variable *test_var*. If only one test value is specified, the dimensionality is reduced by one.

In the second method, *file* is a netCDF file which is structurally identical to *infile* except that the dependent/independent relationship of a variable and a dimension have been reversed. The new, independent, coordinate variable in *file* (corresponding to the old, dependent variable in *infile*) should have values at those positions to which the old, independent dimension of *infile* should be interpolated, supplanting whatever values exist in *template*. Fill values in *template* supplant those in *infile*.

If the dependent/independent relationship isn't strictly invertible, `min` and `max` specify, respectively, that the interpolated value should be the least or greatest possible one.

As an example, values of meteorological fields on isentropic surfaces (surfaces of equal potential temperature) can be obtained by invoking `ncisocut` with *dimname* equal to the pressure or height dimension, *test_var* equal to the name of the potential temperature field, and *test_val1*,*test_val2*,... equal to the desired values of the potential temperature.

### 2.6 ncselect

**`ncselect [-o]`** *condition infile outfile*

Selects data that satisfies a specifiable condition. Variables for which the condition is meaningful have values that satisfy the condition copied to the output file. *Condition* is a relational expression involving dimensions, variables, and constants. Acceptable conditions include expressions like `rec > 20` for an unlimited rec dimension and `T < 5.0 && z < w` for variables `T`, `z`, and `w`. The condition must be quoted if it involves shell metacharacters such as < or >. The `-o` option specifies that only variables affected by the condition shall be copied (the default is to copy all variables for which the conditions is meaningless).

## 3.0 Combiners

These operators accept multiple input files from which data is combined in various ways to form an output file. Some of these use the concept of **conformable** netCDF files, which are files that have the same structure (variables, dimensions, and attributes) but possibly different data and attribute values.

### 3.1 ncappend

**`ncappend [-d`** *dimname***`]`** *infile1 infile2 ... outfile*

Laminates conformable netCDF files by creating a record dimension, *dimname*, increasing the dimensionality of each input variable by one for the new record dimension. If the output file does not already exist, it is created and will have a record dimension named *dimname*. If the input variables each have an attribute named *dimname*, and that value is the same for all variables in a file, then its value is used for the record dimension coordinate variable; otherwise, the file position is used. It is permissible to specify as the record dimension name the name of a scalar variable in the input files, in which case this variable will become a coordinate variable with values taken from each input file, in the order in which they occur on the command line. All input files must have the same structure.

If the output file exists, then the `-d` option is unnecessary: the data in the input files is appended to it using the existing record dimension. If *dimname* is given, then it must be the same as the unlimited dimension name in the output file. Every input file must have the same structure as each record of the output file.

This operator is the inverse of `ncrdim`.

## 3.2 nccat

**`nccat [-d`** *dimname***`]`** *infile1 infile2 ... outfile*

Merges conformable netCDF files by concatenation in the specified dimension (or in the unlimited dimension if no dimension is specified). The input files must have the same structure except that they may have different sizes along the concatenation dimension. It often makes sense to follow an `nccat` operation with an `ncsort` operation along the concatenation dimension, to obtain an output file with monotonic coordinate values along that dimension.

## 3.3 ncmerge

**`ncmerge [-s`** *suf1***`[,`***...***`]]`** *infile1 infile2 ... outfile*

Merges possibly dissimilar netCDF files into a single file. Variable and global attribute name clashes are avoided by appending numeric suffixes corresponding to input-file command-line position. Alternatively, a list of suffixes may be specified after the `-s` option. In this case, one suffix must be specified for each input file, and the suffixes must be unique.

If clashes occur in dimension names, consistency is checked in sizes and values of the coordinate variables. If there is consistency, only one set of coordinate variables and dimensions is kept. Unlimited dimensions after the first encountered one are turned into ordinary dimensions. Otherwise, automatic suffixes are added as for other variables. In this way variables from dissimilar files defined over identical spaces may be reasonably combined.

This operator is for exchanging “kitchen sink” information.

### 3.4 ncpaste

**`ncpaste`** *infile1* **`[`***infile2 ...***`]`** *outfile*

Forms the union of the input netCDF files. Variables having the same name and coordinate positions are merged, with valid data values in later input files supplanting those in earlier ones.

This operator is the inverse of `nccut`.

### 3.5 ncjoin

**`ncjoin`** *infile1* **`[`***infile2 ...***`]`** *outfile*

Forms the natural join of the input files. The join is formed using as key columns those dimensions which are common to all input files. Such dimensions are called **join-dimensions**. Key values for each join-dimension are taken from the dimension's coordinate variable if it exists in all input files; otherwise, the dimension index is used. Variables not defined over join-dimensions will not appear in the output. Note that it is possible to join multi-dimensional elements. For example, vectors of temperatures from a T(p,t) array can be joined with 2-D arrays of heights from a z(x,y,t) array via their common time dimension, *i.e.*

$$\{T(p,t), z(x,y,t)\} = T(p,t) \otimes z(x,y,t)$$

## 4.0 Mathematical Operators

This category contains a variety of different operators..

### 4.1 ncinterp

**`ncinterp`** *infile1 infile2 outfile*

Linearly interpolates data in the first netCDF input file to the grid point locations of the second. Only common dimensions (*i.e.* ones with the same name in both files) are used. Variables with no common dimensions are ignored. If a common dimension has a coordinate variable in both files, then the values of the coordinate variable are used in the interpolation; otherwise, the dimension index is used.

### 4.2 ncbarne

**`ncbarne [-r`** *dim,rad***`] [-s`** *rad***`]`** *dev-file pts-file out-file*

This program performs a Barnes analysis in a Euclidean co-ordinate system. *dev-file* contains the deviation data (i.e. perturbations from a mean or first-guess field). *pts-file* specifies the output points at which to estimate the deviation field and may contain other data as well. The analysis dimensions are determined from the intersection of the weighting-function dimensions and the one-dimensional variables in both *dev-file* and *pts-file* (NB: variables in *pts-file* corresponding to the analysis dimensions may be either co-ordinate variables or regular variables). The analysis variables are those in *dev-file* that depend on one or more of the analysis dimensions. *out-file* will contain the results of the analysis. It will be a copy of *pts-file* together with the analysis variables of *dev-file* and their associated attributes. In addition, a variable that estimates the degrees-of-freedom of the estimate at each output point will be created for each analysis variable. Its name will be that of the associated analysis variable with the suffix '`_df`'.

There are constraints on the netCDF data-types. The variables in both *dev-file* and *pts-file* corresponding to the analysis dimensions must be of type `NC_FLOAT`, as must the variables in *dev-file* corresponding to the analysis variables.

The value at an output point is a normalized weighted average of the neighboring deviations:

$$\left(\sum_{j=1}^{npts} w_j y_j\right) / \left(\sum_{j=1}^{npts} w_j\right)$$

The shape of the unnormalized weighting-function at an output point is $w(d) = e^{-d^2}$; where $d^2$ is the square of the distance of the deviation datum from the output point in normalized distance units:

$$d^2 = \sum_{i=1}^{n} \left(\frac{x_i - X_i}{r_i}\right)^2$$

The degrees-of-freedom is estimated by the sum of the squares of the normalized weights:

The `-r` option specifies the length-scales of the weighting-function. It is specified once for each analysis dimension.

The `-s` option specifies the maximum search radius about an output point in normalized distance units. The default value is 2.

## 4.3 ncconvl

**`ncconvl`** *infile kernel outfile*

Convolves a netCDF data file with a netCDF kernel file. Convolution occurs over the dimensions common to both the data and the kernel. Such dimensions are called **convolution dimensions**. Each coordinate-variable corresponding to convolution dimension must be strictly monotonic and evenly spaced. Such coordinate-variables that exist in both the data and kernel must also agree in grid spacing and monotonicity.

### 4.4 ncarith

**ncarith** *result***=***expression* ... *infile outfile*

Performs point by point arithmetic. Each expression is in the form of a Standard C language expression (or some subset, if this is too hard). The variable on the left side of the equals sign may or may not already exist in the file. If it doesn't, it is created with a dimensionality that is the union of the dimensionality of all the input variables. If the output variable is already defined, then it must have this dimensionality. Constants and function calls may occur in each expression. Desired functions are, at a minimum, `sin`, `cos`, `tan`, `atan2`, `exp`, `log`, `log10`, and `pow`, as defined in the Standard C math library. Coordinate variables must not be changed.

### 4.5 nccalc

**nccalc -r | -i**
**-v** *var1[,...]*
**-d** *dim1[,...]*
**-r** *res1[,...]* *infile outfile*

Performs calculus operations on variables *var1*,.... Either integrals (`-i`) or derivatives (`-r`) are done for each variable over the specified dimensions *dim1*,.... The results are stored in new variables with names *res1*,....

## 5.0 Miscellaneous

### 5.1 ncstat

**ncstat [-d** *dim1***[,**...**]] [-s** *stat1***[,**...**]]** *infile outfile*

Creates a netCDF file that contains a summary of an input netCDF file over the specified dimensions, *dim1[,...]*, or over all dimensions if the `-d` option is not used. The specified summary operations include `min`, `mean`, `max`, and `fillcount` for the minimum, average, and maximum values and for a count of the number of fill values of each variable over the specified dimensions. If no operations are specified with the `-s` option, all statistics are assumed. Variables that do not involve the selected dimensions are copied to the output. Other variables that use the specified dimensions are replaced by summary variables with reduced dimensionality.

As an example, if the input contains a variable named `var(i,j,k)` and `-d i,k` is specified, then the output netCDF file will have the variables `var_max(j)`, `var_min(j)`, `var_mean(j)`, and `var_fillcount(j)`.

The `*_max`, `*_min`, and `*_mean` variables inherit all the attributes of the variable from which they are derived; first two also inherit the same type. The `*_mean` variable is of type `float` unless the original variable is of type `double`, in which case it is also of type `double`. The `*_fillcount` variable is always of type `long` and inherits no attributes from the original variable.

If no dimensions are specified (*i.e.* all dimensions are selected) then the output netCDF file will have four scalar variables for each variable, *var*, in the input: *var*`_max`, *var*`_min`, and *var*`_mean`, and *var*`_fillcount`, except that scalar variables in the input are merely copied. Fill values do not participate in the computation of minimums, maximums, or means.

## 5.2 ncsort

**ncsort** **[-v** *name*] [...] **[-ru]** *infile outfile*

Sorts data according to the value of variables (including coordinate variables). Numeric sort variables must be one-dimensional. Character sort variables may be two-dimensional, in which case they will behave like one-dimensional string variables for the purpose of sorting. If no variables are specified, then records are sorted along the unlimited dimension. If a given variable does not exist, but a dimension of the same name does, then the variable is taken to be the dimension index; hence, an increasing sort on such a variable will merely copy its input to output. By default, data is sorted in order of increasing values of the named variables, with the first variable being the primary sort key, the second variable being the secondary sort key, *etc*. This order is changed to decreasing by the `-r` option. If the `-u` option is present, then duplicate values of the sort dimension (together with the associated data) are not copied to the output — resulting in a strictly monotonic variable. String variables are sorted using the native collating sequence of the operating system.

Some netCDF operators require that certain variables be monotonic or even strictly monotonic.

## 5.3 ncrename

**ncrename** **[-d** *old*,*new*] **[-d** ...]
**[-v** *old*,*new*] **[-v** ...]
**[-a** *old*,*new*] **[-a** ...] *infile outfile*

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 in the input file, unless the name is preceded by the character '`.`'. If the output file is the same as the input file, renaming is done "in-place". Note that this will still incur the overhead of copying the whole file if any of the new names are longer than the old names.

Note that renaming a dimension to the name of a dependent variable can be used to invert the relationship between an independent coordinate variable and a dependent variable. In this case, the named dependent variable must be one-dimensional and should have no missing values. Such a variables will become a coordinate variable.

## 5.4 ncorder

**ncorder [-n] -d** *dim1***,**... *infile outfile*

Changes the order of dimensions in all variables to that specified on the command-line (the leftmost command-line dimension varies least rapidly). All dimensions must be specified. Variables that are defined using more than one dimension and in an order that differs from the command-line order, will undergo a structural reorganization. The new first dimension will become the new unlimited dimension, unless the `-n` option is present, in which case the output file will have no unlimited dimension.

## 5.5 ncunpack

**ncunpack** *infile outfile*

Uncompresses limited-precision floating-point data which is packed as small integers using the `scale_factor` and `add_offset` attributes. The type of the resulting unpacked variable is determined by the highest type of its `scale_factor` and `add_offset` attributes.

If a `_FillValue` attribute exists for a packed variable, then it is propagated, through unpacking, into a `_FillValue` for the expanded type.

This is the inverse of the `ncpack` operator.

## 5.6 ncpack

**ncpack [-b** *var1***[,**...**]] [-s** *var1***[,**...**]]** *infile outfile*

Compresses floating-point data into small integers. If no variable lists are specified, all floating-point and double-precision variables are packed into 8-bit, unsigned integers. Otherwise, only the specified variables are packed into either 8-bit (`-b`) or 16-bit (`-s`) unsigned integers. For each packed variable, the `scale_factor` and `add_offset` attributes are used. If these attributes exist in the input file, then their values control the packing; otherwise, they are automatically created in the output file and have same type as the unpacked variable data.

If a `_FillValue` attribute exists for a variable, then it is propagated, through packing, into a `_FillValue` for the compressed variable.

This is the inverse of the `ncpack` operator.

## 5.7 nccmp

**nccmp [-r** *releps***] [-a** *abseps***]** *infile1 infile2*

Compares two netCDF files and reports any differences. If *releps* is specified, then floating point numbers, *x* and *y* (in *infile1* and *infile2*, respectively) that satisfy

$$abs\,(x - y) > releps \times max(abs(x), abs(y))$$

will be considered different. If *abseps* is specified, then floating point numbers that satisfy

$$abs\,(x - y) > abseps$$

will be considered different. If both *releps* and *abseps* are specified, only *x*, *y* pairs that satisfy both criteria will be considered different. If neither *releps* nor *abseps* is specified, all value differences will be reported.

### 5.8 ncmkgrd

**`ncmkgrd -d`** *name*`,`*npts* **`-v`** *name*`,`*start*`,`*stop*`,`*step* **`[...]`** *outfile*

Generates a netCDF file containing a grid. For each dimension *name*, either the number of grid points, *npts*, or a coordinate-variable specification may be given, but not both. The dimension resulting from a coordinate-variable specification will have *start* as its first coordinate value, *step* as the grid-point spacing, and the last coordinate value will be the one closest to *stop* from *start* in steps of *step*. If *stop* is less than *start*, then *step* should be negative.

## 6.0 Deferred

**View variables** will be supported. View variables are like symbolic links, in that they have no associated data but instead have a special string-valued `_Def` attribute, giving the definition for a variable in terms of a hyperslab subset of another variable in the same or a different file. (Alternatively, if view variables are supported as part of the netCDF interface rather than at this layer, their definition will be part of their creation.) View variables may rename variables, reorder their dimensions, or give names to variable cross-sections. For example,

```
var:_Def = "foo:othervar(lat=(-125.0,-75.0),lon=40.0,
            lvl=(850,400))";
```

defines a two-dimensional view variable `var(lat,lvl)` in terms of a cross-section of a three-dimensional variable `othervar` in another file named `foo` using dimension names associated with the real source variable (which must not clash with local dimension names). Variable attributes and data are inherited from the real variable, except that a view variable may define additional attributes and may override the attributes of its associated real variable. When a view variable is output it is fully instantiated, with data copied into the output netCDF file and appropriate output dimensions created as necessary.

## 7.0 References


1. Fahle, J., *TeraScan Applications Programming Interface, SeaSpace*, San Diego, California, 1989.
2. Raymond, D. J., *Proposal for Netcdf Algebra*, Unidata memo, 1989.
3. Raymond, D. J., "A C Language-Based Modular System for Analyzing and Displaying Gridded Numerical Data," ***Journal of Atmospheric and Oceanic Technology***, **5**, 501-511, 1988.