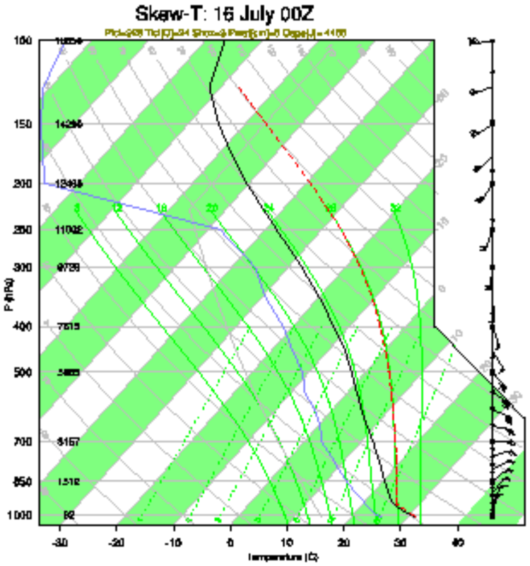
Introduction to NCL

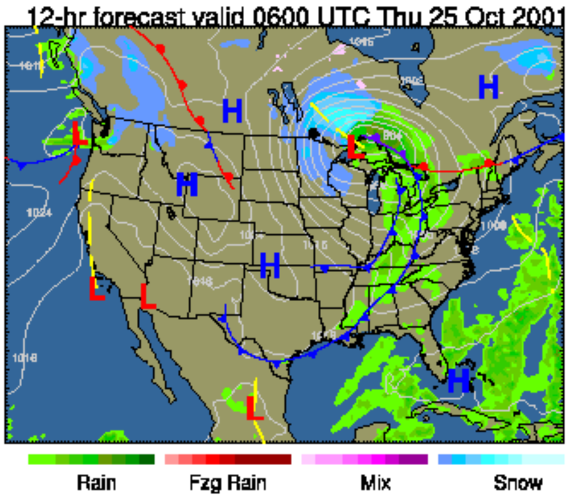
[part 3 of 3]

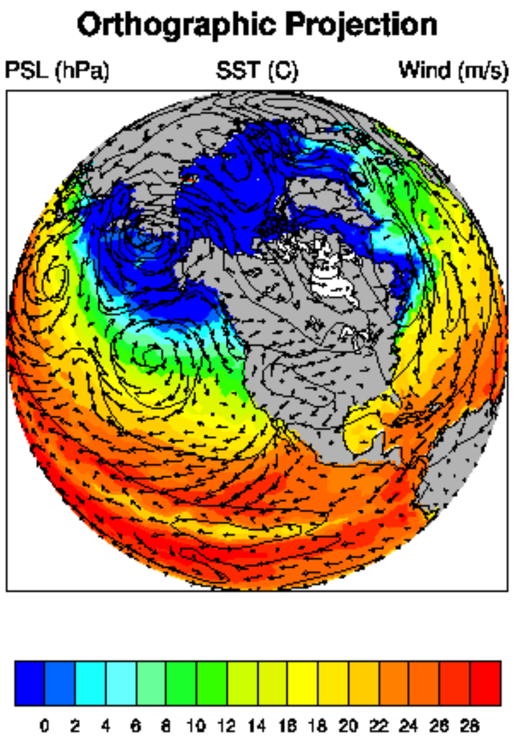
Dennis Shea

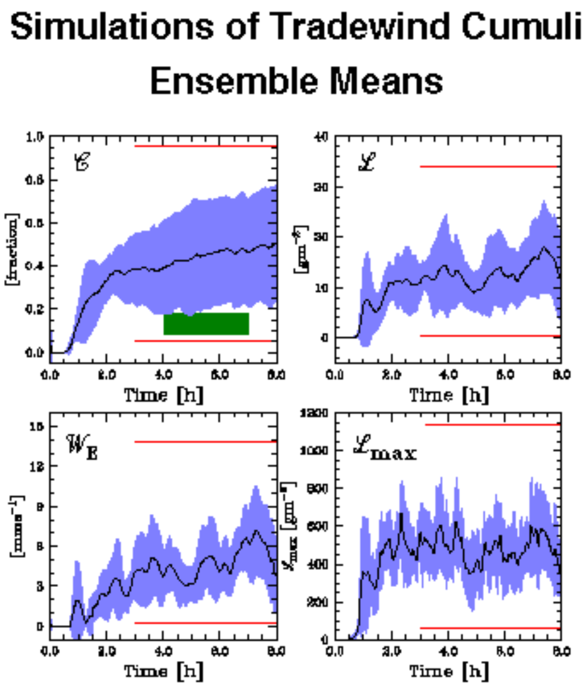
NCAR Command Language

Sponsored by the National Science Foundation









Introduction: Key Points

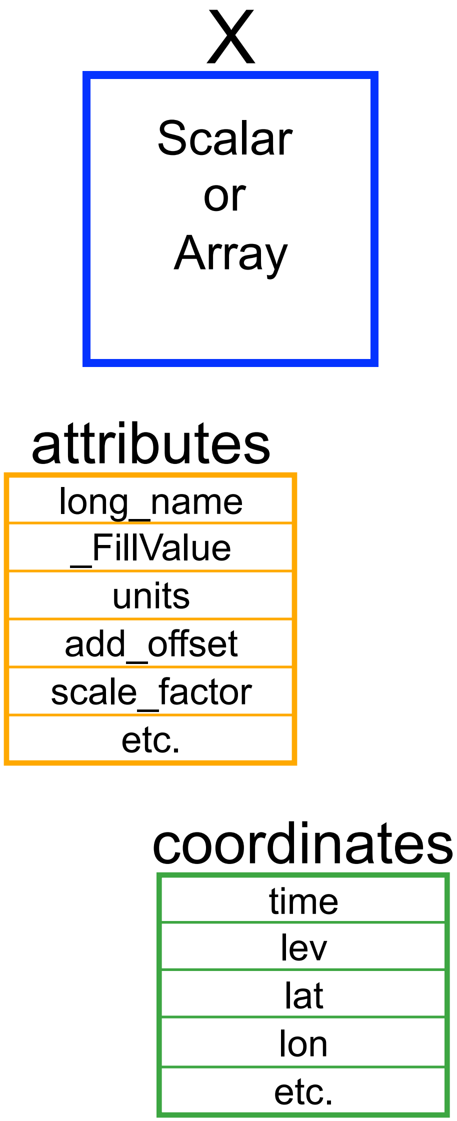
Metadata: information about a variable

- Attributes (**@**)
- Named dimensions (**!**)
- Coordinate variables (&)

Array Subscripting:

- Index (classic integer specification)
- Dimension Numbers (array specification)
- Coordinate Variables (natural coordinates)

netCDF [NCL] Variable model



X

Scalar
or
Array

attributes

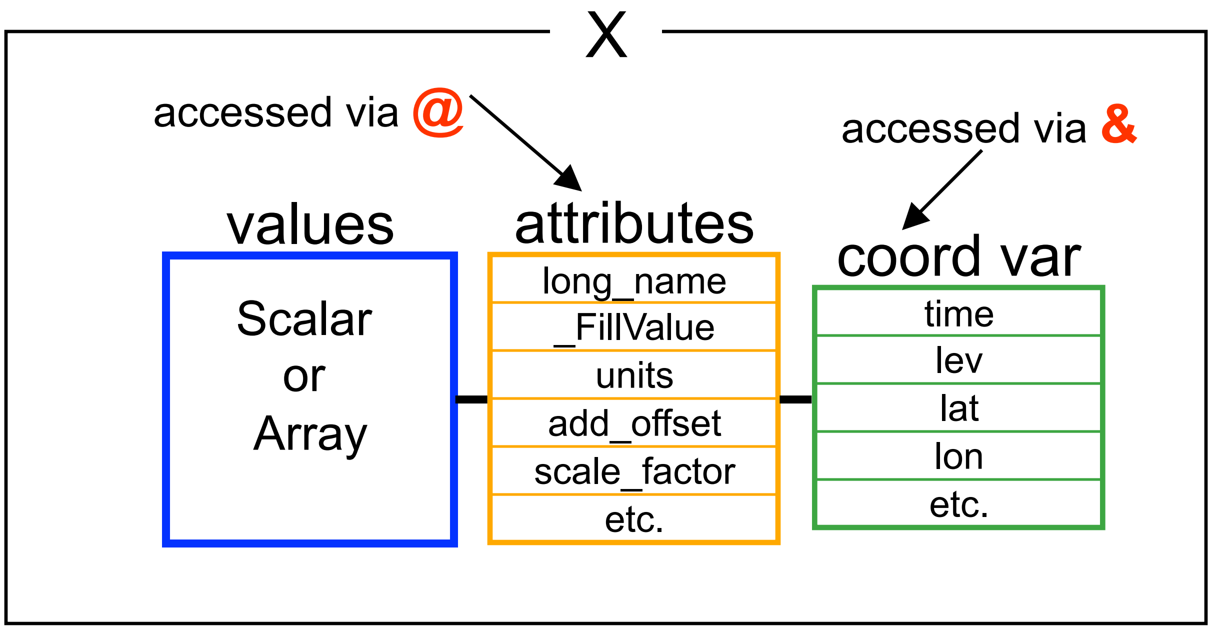
long_name
_FillValue
units
add_offset
scale_factor
etc.

coordinates

time
lev
lat
lon
etc.

```
f = addfile("foo.nc", "r")    ; grb/hdf
x = f->X
```

NCL reads the scalar/array, attributes, and coordinate variables as an object



X

accessed via @

accessed via &

values

Scalar
or
Array

attributes

long_name
_FillValue
units
add_offset
scale_factor
etc.

coord var

time
lev
lat
lon
etc.

Meta Data

- **Information associated with variable or file**
  - numeric or textual
  - meta data is read from files (most frequently)
- **NCL uses syntax to create, retrieve**
  - **attributes: @** (numeric, text)
  - **named dimensions: !** (text)
  - **coordinates: &** (numeric)

Attributes [@]

- **info associated with a variable** or **file**
  - attributes can be any data type except file or list
  - scalar, multi dimensional array (string, numeric)

- **assign/access with @ character**

```
T                  = (/ 10, 25, 39 /)   ; one-dim of length 3
T@units            = “degC”
T@long_name = “Temperature”
T@wgts             = (/ 0.25, 0.5, 0.25 /)
T@x2d              = (/  (/1,2,3/),  (/4,5,6/), (/7,8,9/) /)
T@_FillValue   = -999
title              = T@long_name
```

- **attribute functions** [isatt, getfilevaratts]

```
if (isatt(T,"units")) then ....  end  if
atts  = getfilevaratts (fin, "T")
```

  - **delete** an attribute: **delete**(T@wgts)

_FillValue attribute

- **Unidata & NCL reserved attribute; CF** convention

- netCDF Operators [**NCO & CDO]: _FillValue** attribute
- ncview: recognizes **missing_value** attribute (**COARDS**)
  - best to create netCDF files with both

- **NCL** functions recognize **_FillValue**
  - most functions will ignore for computations (eg, “avg”)
  - use built-in function “**ismissing**” to check for **_FillValue**
  - if (any (**ismissing**(T))) then … end if
    - **NOTE**: if (**any**(T.eq.T@_FillValue)) will **not** work

- Recommendation: do **not** use zero as a **_FillValue**

Arrays: Indexing & Dimension Numbers

- **row major**
  - **left** dimension varies **slowest**; **right** dim varies **fastest**
  - dimension numbering **left to right** [0,1,..]
- **subscripts**
  - **0-based** [entire range for N index values: 0,N-1]

Consider T(:, :, :, :) → T (0, 1, 2, 3)

left	dimension is	**0**	: varies slowest
mid-left	dimension is	**1**	
mid-right	dimension is	**2**	
right	dimension is	**3**	: varies fastest

- Some processing functions operate on dimension numbers
- Example: T(ntim, klev, nlat, mlon) → T(**0, 1, 2, 3**)
  - Tzon = **dim_avg_n**(T, **3**) → Tzon(ntim, klev, nlat)
  - Tstd = **dim_stddev_n**(T, **0**) → Tstd (klev, nlat, mlon)

NCL – Fortran/Matlab/R Array Indexing

Different language/tool ordering. There is no ‘right/wrong’

- **NCL/C/C++** : 0-based; left (slowest) - right (fastest)
- **fortran**, **Matlab**, **R**: 1-based; left (fastest) - right(slowest)
- **IDL** : 0-based; left (fastest) - right(slowest)

- **ncl: x(N,M) => value <= x(M,N) : F/M/R** **M=3, N=2**
  - x(0,0) => 7.23 <= x(1,1)
  - x(0,1) => -12.5 <= x(2,1)
  - x(0,2) => 0.3 <= x(3,1)

  switch to next index

  - x(1,0) => -431.2 <= x(1,2)
  - x(1,1) => 323.1 <= x(2,2)
  - x(1,2) => -234.6 <= x(3,2)

NCL (netCDF): Named Dimensions [!]

- **x(time,level,lat,lon)**
- dimensions are named on netCDF files
  - alternative way to reference subscripts

- **Create (assign) with ! character**
  - T!0 = "time" ; leftmost [slowest varying] dim
  - T!1 = "lat“
  - T!2 = "lon" ; rightmost [fastest varying] dim

- **Dim names may be renamed, retrieved**
  - T!1 = ”latitude" … dName = T!2
- can delete/eliminate: delete (T!2)

- **Named dimensions used to reshape**
  - **x**(**lat**|:, **level**|:, **lon**|:, **time**|:)

Create, Assign Coordinate Variables [&]

- **create 1D array**
  - time = (/ 1980, 1983, 1994 /)
  - time@units = “yyyy”
  - lon = **ispan**(0, 355, 5)
  - lon@units = “degrees_E”

- **assign dimension name** [same as variable name]
  - time!0 = “time”
  - lon!0 = “lon”

- let **x**(:,:) … dimension numbers x(0,1)
- **name dimensions**
  - x!0 = “time” … x!1 = “lon”
- **assign coordinate variables to x**
  - x&time = time … x&lon = lon

Meta Data Syntax Review: Access/Change/Create/Delete

- @ attribute
  - u@long_name = "U"
  - lonName = u@long_name

- ! named dimensions
  - u!0 = "TIME"
  - tName = u!0

- & coordinate variable
  - u&lat = (/ -90., -85, ... , 85., 90. /)
  - latitude = u&lat

- $ substitute string
  - x = fin->$variable(n)$ ... x = fin->$"T: p"$

Variable Subscripting (1 of 3)

Standard Array Subscripting (Indexing)

- index: start:end [:optional stride]; iStrt:iLast:iStride
- index values separated by a colon :
- omitting start/end index implies default begin/end

Consider T(time,lat,lon)

T	→	entire array [don't use T(:,:,:)]
T(0,:,::5)	→	1st time index, all lat, every 5th lon
T(1:3, ::-1, :50)	→	3 time indices, reverse, 1st 51 lon
T(7:12,45,10:20)	→	6 time indices, 46th value of lat, 10-20 indices of lon

Programming tip: **use variables not hard wired #**

T(**tstrt:tlast**, **:** , **ln1:ln2**) → time index **tstrt:tlast**, all lat **:**,
longitude index values **ln1:ln2**

Variable Subscripting (2 of 3)

Coordinate Variable Subscripting

- **only** applies to coordinate variables (1D, mono)
- same rules apply for ranges, strides, defaults
- use curly brackets **{…}**
- standard and coordinate subs can be mixed [if no reorder]

T(2:7,**{**-30:30**}**,:) ➔ six times, all lon, lat -30° to +30° (inclusive)

T(0,**{**-20**}**,**{**-180:35:3**}**) ➔ 1st time, lat nearest -20°, every 3rd lon between -180° and 35°

T(::12,**{latS:latN}**,:) ➔ all times/lon, lat latS to latN (inclusive)

T(8,**{latS}**,**{lonL:lonR**:3**}**)➔ 9th time, lat nearest **latS**, every 3rd lon between **latL** and **lonR**

Variable Subscripting (3 of 3)

Named Dimensions

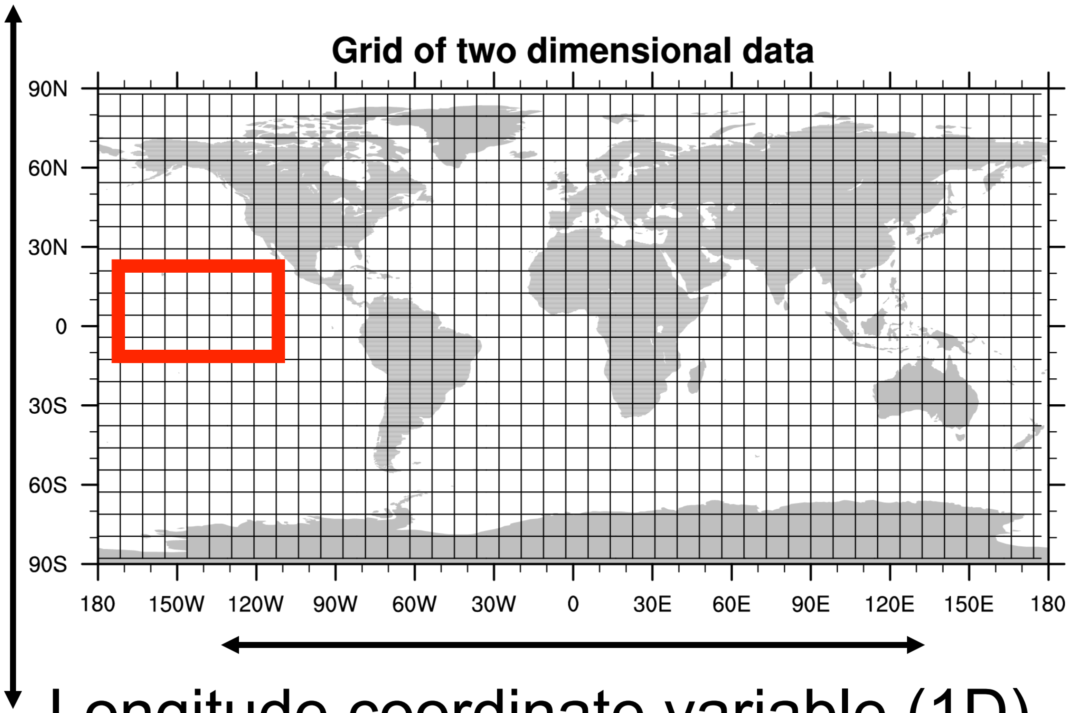
- **only** used for dimension reordering
- indicated by |
- dim names must be used for each subscript
- named/coordinate subscripting can be mixed

Consider T(time,lat,lon)

```
t = T(lat|:, lon|:, time|:)                ➔ makes t(lat,lon,time)
t = T(time|:, {lon|90:120}, {lat|-20:20}) ➔ all times,
                                             90-120° lon, -20-20° lat
```

Subscripting: Index, CV

Latitude coordinate variable (1D)



Longitude coordinate variable (1D)

Standard:

T(**9:13**,**1:8**)

Coordinate:

T(**{-10:20}**,**{-170:-110}**)

Combined:

T({-10:20}, 1:8)