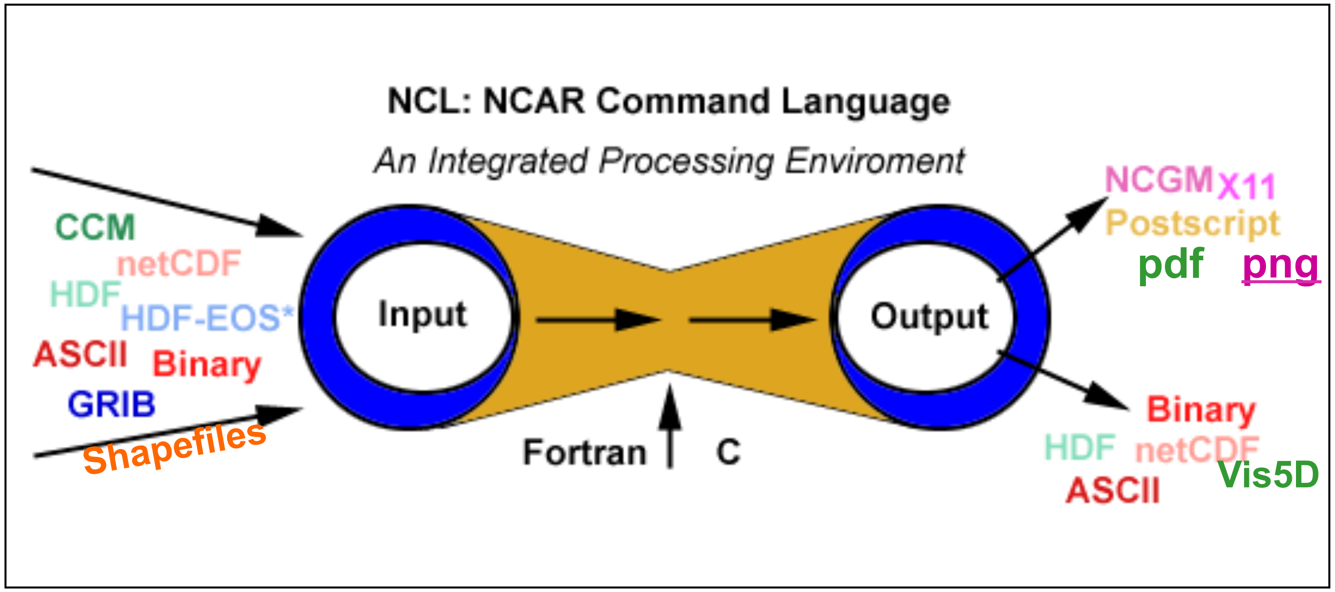
File Input/Output

Supported Formats



Dennis Shea

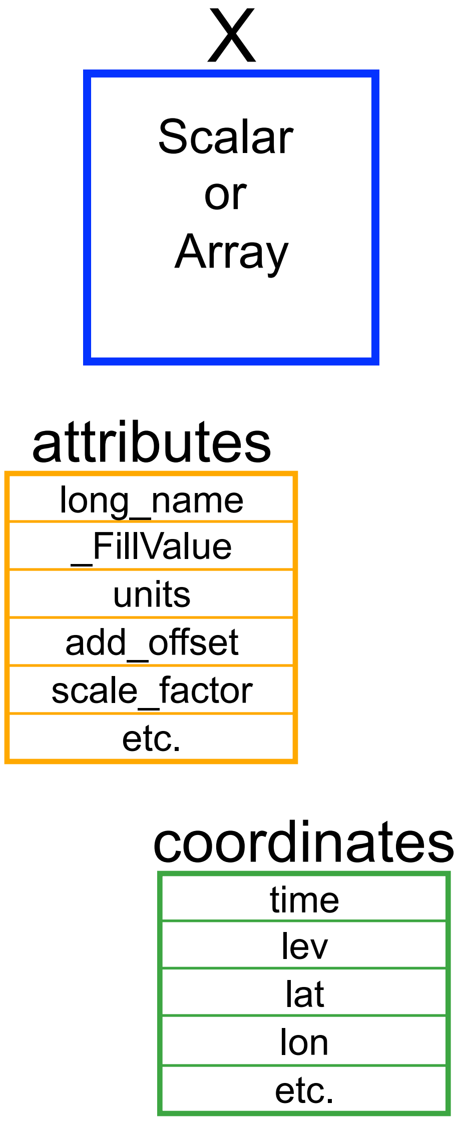
National Center for Atmospheric Research

NCAR is sponsored by the National Science Foundation

NCL Supported Formats

- **Supported** formats
  - User need **not** know internal structure of files
- **Formats**
  - **netCDF-3/4** [network Common Data Form]
  - **HDF4/H5** [Hierarchical Data Format]
  - **HDF-EOS** [Earth Observing System; HDF4 and HDF5]
  - **GRIB-1/2** [GRId in Binary; WMO standard; NCEP, ECMWF,…]
  - **CCMHT** [CCM History Tape; COS blocked only; ccm2nc]
  - **Shapefile** [ESRI: geospatial vector data format GIS]
  - **6.2.1** ➔ near complete netCDF4, HDF5
- **GRIB**
  - 50+ lookup tables builtin which provide meta data
  - latitude/longitude arrays created
- **Command line operators for supported formats**
  - Utilities to provide file overview; change format
- **Users need not 'fear' any Supported Format**
  - NCL imports variables into a **common data structure**

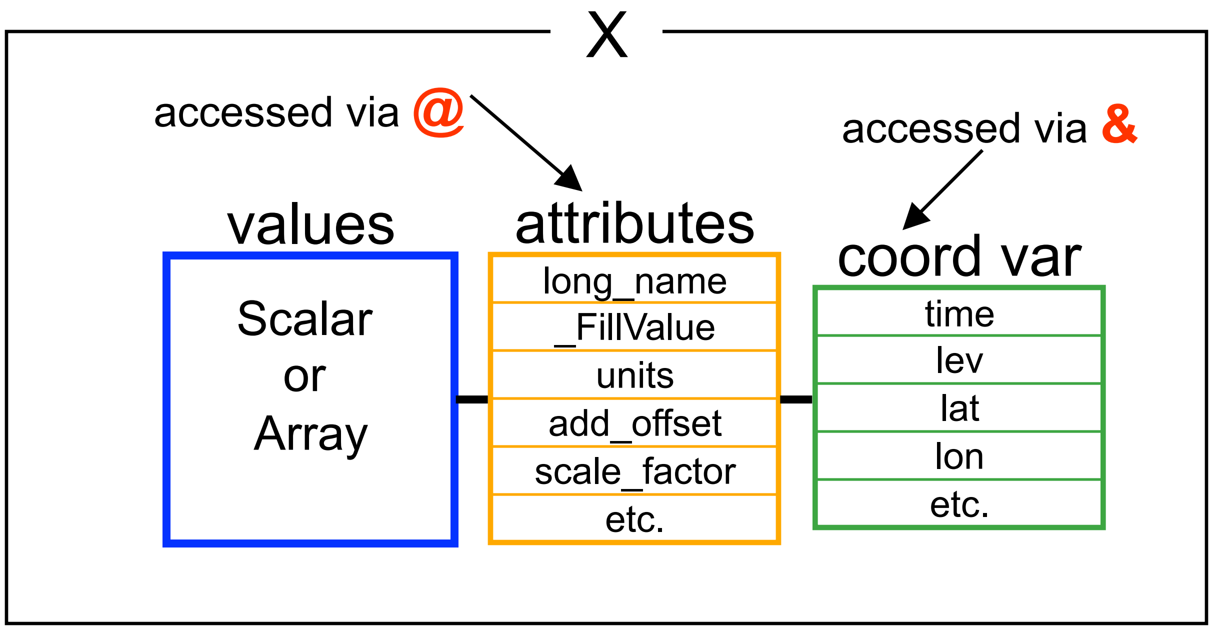
netCDF [NCL] Variable model



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

x = f->X
```

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



ncl_filedump

http://www.ncl.ucar.edu/Document/Tools/ncl_filedump.shtml

- **ncl_filedump [-c] [-v var1[,…]] [–h] file_name**
  - command line utility with **options**
  - provides textual overview of **any supported** file's contents
  - behavior analogous to Unidata's **ncdump -h**
  - **file_name** must have a file type suffix on command line
    - **.nc .grb .hdf .hdfeos .he5 .h5 .ccm .shp** *[case insensitive]*
    - suffix used as identifier only, actual file need not have

- **ncl_filedump file_name.[grb/nc/hdf/hdfeos]**
  - output can be sent to file or viewer via Unix redirection/ pipe

```
ncl_filedump foo.grb > foo.txt        [send to file]
ncl_filedump foo.hdf  | less          [send to viewer]
```

ncl_convert2nc
http://www.ncl.ucar.edu/Document/Tools/

- **ncl_convert2nc gribFile(s)** ***OPTIONS***
    - command line utility
    - converts GRIB/HDF/SHAPE file(s) to netCDF
    - output name same as input with **.nc** extension
- **ncl_convert2nc –h**
    - provides usage option information
- **ncl_convert2nc foo.grb**
    - will create **foo.nc**
- **ncl_convert2nc foo.hdf –L –nc4c –cl 1**
    - **-L** (files> 2GB); **-nc4c** (netCDF4); **-cl 1** (compression lvl 1)

setfileoption

www.ncl.ucar.edu/Document/Functions/Built_in/setfileoption.shtml

- **allows user to specify file-format-specific options**
  - netCDF, GRIB and Binary options *[currently]*

- **sample usage of selected options**
  - writing netCDF
    - **setfileoption**(“nc”, "DefineMode" ,True)
    - **setfileoption**("nc","Format","LargeFile")
    - **setfileoption**("nc","Format",”netCDF4Classic")
  - reading GRIB
    - **setfileoption**("grb" ,"ThinnedGridInterpolation", "cubic")
    - **setfileoption**("grb", "InitialTimeCoordinateType" \
      , "Numeric”)

    setfileoption("grb", "TimePeriodSuffix” ,False)

addfile (1 of 3)

- Opens a **supported** format
- Variables look like netCDF (Grib, HDF, HDF-EOS)

- **f** = **addfile** (**file_name.ext**, **status**)
  - **file_name** => any valid file name; string
  - **ext** => extension that identifies the type of file; string
    - netCDF: **"nc"** or **"cdf"** [read/write]
    - HDF: **"hdf"** , **"hdfeos"**, **"h5"**, **"he5"** [read/write]
    - GRIB: **"grb"** , **"grib"** [read only; GRIB1 or GRIB2]
    - CCMHT: **"ccm"** [read only]
    - SHAPE (GIS): **"shp"** [read only]
    - extension **not** required to be attached to file
  - **status** [read/write status] **"r"**, **"c"**, **"w"**
  - **f**
    - reference/pointer to a single file; any valid variable name
    - may have attributes (file attributes or global attributes)

http://www.ncl.ucar.edu/Document/Manuals/Ref_Manual/NclFormatSupport.shtml

addfile (2 of 3)

- **Examples: opening a single file**
  - fin   = addfile ("0005-12.nc"   , "r")
  - fout = addfile ("./ncOutput.nc" , "c")
  - fio   = addfile ("/tmp/shea/sample.hdf"   , "w")
  - g   = addfile ("/dss/dsxxx/Y12345.grb", "r")
  - s   = addfile ("foo.shp" , ”r")

- **Numerous functions to query contents of supported file**
  - getfilevarnames
  - getfilevardims
  - getfilevaratts
  - getfilevardimsizes
  - getfilevartypes
  - presentvar
  - isfilevaratt
  - isfilevardim
  - isfilevarcoord

```
diri  = "/fs/cgd/data0/shea/GRIB/"
fili   = ”narr_2000121106”
fin    = addfile(diri+fili+".grb" , " r ")
```

```
varNames = getfilevarnames (fin)
if (isfilevarcoord(fin, "U", "lat") ) then
…
end if
```

addfile: OPeNDAP (3 of 3)

- **OPeNDAP enabled:** Open Source Project for Network Data Access Protocol
  - access a remote file over the internet
  - file must be located on an OPeNDAP server [max 64 files]
  - only certain operating systems are currently OPeNDAP enabled. NCL can perform OPeNDAP operations on supported systems. Some (CDC) require registration.
  - works with addfile, addfiles, and isfilepresent

```
url_cdc = "http://www.cdc.noaa.gov/cgi-bin/opendap/nph-nc/Datasets/"
fPath    = “ncep.reanalysis/pressure/air.1948.nc"
if ( isfilepresent(url_cdc+fPath) ) then
     f              = addfile ( url_cdc + fPath, "r")
     vNames = getfilevarnames(f)
     if ( any (vNames .eq. ”T”)) then
          t = f->T
     end if
end if
```

Import Variable from Supported Fmt

u = f->U

- read **variable** and **all** meta data into memory **[structure]**
- no space allowed to left/right of **->** [fatal error]
- use **"$"** syntax to represent variable name if type string

```
f = addfile ("foo.grb", "r")
vNam = getfilevarnames (f)  ; all variables on file
        or
vNam = (/ "SLP", "T" /)        ; manually specify
do n=0,dimsizes(vNam)-1
    x := f->$vNam(n)$          ; $..$ substitute string
     .....
end do
```

u = (/ f->U /)

- read data values only and _FillValue attribute

Example: open, read, output netCDF

```
begin          ; optional
;---------------------------------------------------
   fin    = addfile ("in.nc, "r")      ; open file and read in data

   u      = fin->U                     ; import a variable (time,lev,lat,lon)

   fout   = addfile("out.nc" , "c") ; create reference to output file

   fout@title = "I/O Example 1"  ; add a global attribute to the file

;---------------------------------------------------
;Output variable :    ncrcat/ncks  –v  U in.nc out.nc
;---------------------------------------------------
    filedimdef (fout, "time", -1, True)  ; create unlimited dim
    fout->U2 = u                         ;  output variable u to nc file
end                                      ; only if begin is present
```

Note: This method of netCDF creation has simple syntax. It can be slow but is commonly used.

Example: query file, system commands

```
;-----------------------------------------------------------------
; open file, create array of variable names, # of names
;-----------------------------------------------------------------
       fin     = addfile ("./in.nc", "r")
       vars   = (/"U", "V", "T" /)           ; manual specification
       nvars = dimsizes (vars)               ; nvars = 3
;-----------------------------------------------------------------
; use system to remove output file before creation
;-----------------------------------------------------------------
       fname = "out.nc"
       system("/bin/rm –f   "+fname)
       fout     = addfile(fname, "c")
;-----------------------------------------------------------------
; loop, query if variable on the file, then output to netCDF
;-----------------------------------------------------------------
       do n=0,nvars-1
              if (isfilevar(fin, vars(n)))  then
                     fout->$vars(n)$ = fin->$vars(n)$
              end if
       end do
```

```
ncrcat/ncks  –v  U,V,T   in.nc   out.nc
```

Import byte/short Variable (1 of 2)

```
us = f->U   ; read variable and meta data into memory
```

```
Variable: us
Type: short                                  byte
Total Size: 1429632 bytes           147456 bytes
              714816 values          714816 values
Dimensions and sizes:  [time | 4] x [lev |17] x [lat | 73 ] x [lon |144 ]
Number of Attributes: 4
              long_name: zonal wind component
              units:          m/s
              scale_factor:  0.15              [slope: 0.15]
              add_offset:    -3.0              [intercept: -3.0]
```

(generally) user wants to convert to float

- **COARDS** convention: scale value then add offset

```
uf = us*us@scale_factor + us@add_offset
```

better to use contributed.ncl [short2flt, byte2flt]

```
u = short2flt(f->u)    ;    u = byte2flt(f->u)
```

Import byte/short Variable (2 of 2)

(often) HDF files do not conform to COARDS

- add offset to value then scale

```
uf = ( us + us@add_offset )*us@scale_factor
```

better to use contributed.ncl [short2flt_hdf, byte2flt_hdf]

```
u = short2flt_hdf(f->u)    ;    u = byte2flt_hdf(f->u)
```

Simple netCDF [hdf] Creation

```
fout           = addfile ("foo.nc", "c")
fout@title  = "Simple Example"
fout@creation_date = systemfunc("date")
                                          ; if 'time'
filedimdef (fout, "time", -1, True)   ;  create ud

fout->U      = u
fout->T       = Temp
```

- commonly used
- **writes all variable components** [data object ;-)]
- may be inefficient (**possibly**, very inefficient)
- use for file with few variables/records

Efficient netCDF Creation

- **requires ‘a priori’ definition of file contents**
  - must be done in other languages/tools also [F, C, IDL, ..]

- **NCL functions to predefine a netCDF/HDF file**:
  - **setfileoption**: specify entering define mode
  - **filevardef**: define name(s) of one or more variables
  - **filevarattdef**: copy attributes from a variable to one or more file variables
  - **filedimdef**: defines dimensions including unlimited
  - **fileattdef**: copy attributes from a variable to a file as global attributes

- Less tedious than other languages

Example: Efficient netCDF Creation

```
   T      = .....
   fout   = addfile("out.nc" , "c")
   setfileoption (fout, "DefineMode",True))   ; enter define mode

; create global attributes
   fileAtt                     = True
   fileAtt@creation_date = systemfunc("date")
   fileattdef (fout, fileAtt)
; predefine coordinate variables
   dimNames          = (/ "time", "lat",  "lon"/)
   dimSizes          = (/      -1 ,  nlat, mlon/)   ; -1 means unknown
   dimUnlim          = (/  True , False, False/)
   filedimdef (fout, dimNames, dimSizes, dimUnlim)
; predefine variable names, type, and dimensions
   filevardef  (fout, "time", typeof(time), getvardims(time))
   filevardef  (fout, "lat"   , typeof(lat)   , getvardims(lat)   )
   filevardef  (fout, "lon"   , typeof(lon)   , getvardims(lon)   )
   filevardef  (fout,"TMP" , typeof(T)     , getvardims( T )   )
; create var attributes for each variable
   filevarattdef (fout, "TMP", T)

; output data values only  [use  (/… /) to strip meta data]
   fout->time    = (/ time /)
   fout->lat      = (/ lat    /)
   fout->lon      = (/ lon    /)
   fout->TMP   = (/  T      /)    ; note the different name on file
```

Contents of a well written netCDF variable

- **Variables**
  - **long_name***
  - **units***
  - **_FillValue** [if applicable]
  - missing_value [“]
  - named dimensions
  - coordinate variable(s)

```
          Consider: T(:)
T@long_name      = "Temperature"
T@units          =  ”degC"
T@_FillValue     = 1e20
T@missing_value = T@_FillValue
T!0 = "time"
T&time = time
        Result:   T(time)
```

*__COARDS__ and **CF** conventions

CF Compliance Checker:

http://puma.nerc.ac.uk/cgi-bin/cf-checker.pl

Importing Multiple Supported Files

- **systemfunc:** returns info from unix/linux
  - fnames = **systemfunc** ("**ls** reAnal*")
    - fpath = **systemfunc**("**ls** /mydata/reAnal*") ; full path
    - fils   = **systemfunc**("**cd** "+path+ " **; ls** reAnal*")

      where: path = "/my/data/"
- **manually**
  - fnames = (/ "file1" , "file2", ... /)

```
diri          = “/data0/shea/"
fili          = (/ "reAnal1", "reAnal2", "reAnal3", "reAnal4"/)
nfiles        = dimsizes(fili)                  ; nfiles = 4
do nf =0,nfili-1
   f = addfile (diri+fili(nf)+".grb", "r")
   ……
end do
```

addfiles (1 of 2)

- span **multiple supported** files

- **q** = **addfiles** (**fNames**, "r")
  - **fNames** is a 1D array of file names (strings)
  - can be used for **any supported format**
  - technically, "q" is a variable of type **list**

```
T = q[:]->T                ; [:] read all files
```

- read T [with meta data] from each file in list 'q'
- T must exist in each file and be same shape [rank]
- a **list** is used to sequence results of **addfiles**
- normal file variable selection is used with "**[…]**"

```
lat = q[0]->lat        ; [0] read from first file
Z  = q[2:6:2]->Z   ; extract Z only from files 2,4,6
```

addfiles (2 of 2)

- 2 options on variable merging
  - **ListSetType** (a, **"cat"**) [default; "cat" => concatenation]
  - **ListSetType** (a, **"join"**)

- when to use "cat" and "join" [rule of thumb]
  - **cat**: continuous record
  - **join**: creating ensembles
    - a record dimension will be added

netCDF Operator (NCO): **cat** ➔ **ncrcat** **join** ➔ **ncecat**

Example: Read "T" across 5 files ["cat"]
[Each file has 12 months]

```
fils   = systemfunc ("ls  ./ann*.nc")
f       = addfiles (fils, "r")
ListSetType(f, "cat")            ; not necessary [default]
T     = f[:]->T                   ; read T from all files
printVarSummary(T)
```

```
Variable: T
Type: float
Total Size:  5529600 bytes
             1382400 values
Attributes: 2
     units:        K
     long_name: temp
Number of Dimensions: 4
Dimensions and sizes: [time|60] x [lev|5] x [lat | 48]  x  [lon | 96]
Coordinates:
time: [2349 … 4123]      lat:    [-87.159..87.159]
lev:   [85000 … 25000]   lon:   [0..356.25]
```

addfiles: option ["join"]

```
fils  = systemfunc ("ls ./ann*.nc")
f     = addfiles (fils, "r")
ListSetType (f, "join")
T    = f[:]->T
printVarSummary (T)
```

```
Variable: T
Type: float
Total Size:  5529600 bytes
             1382400 values
Attributes: 2
   units:         K
   long_name: temperature
Number of Dimensions: 5
Dim/sizes:  [case | 5] x  [time|12]  x  [lev|5]  x  [lat | 48]  x  [lon | 96]
Coordinates:
time: [2349 … 2683]      lat:    [-87.159..87.159]
lev:   [85000 … 25000]   lon:    [0..356.25]
```