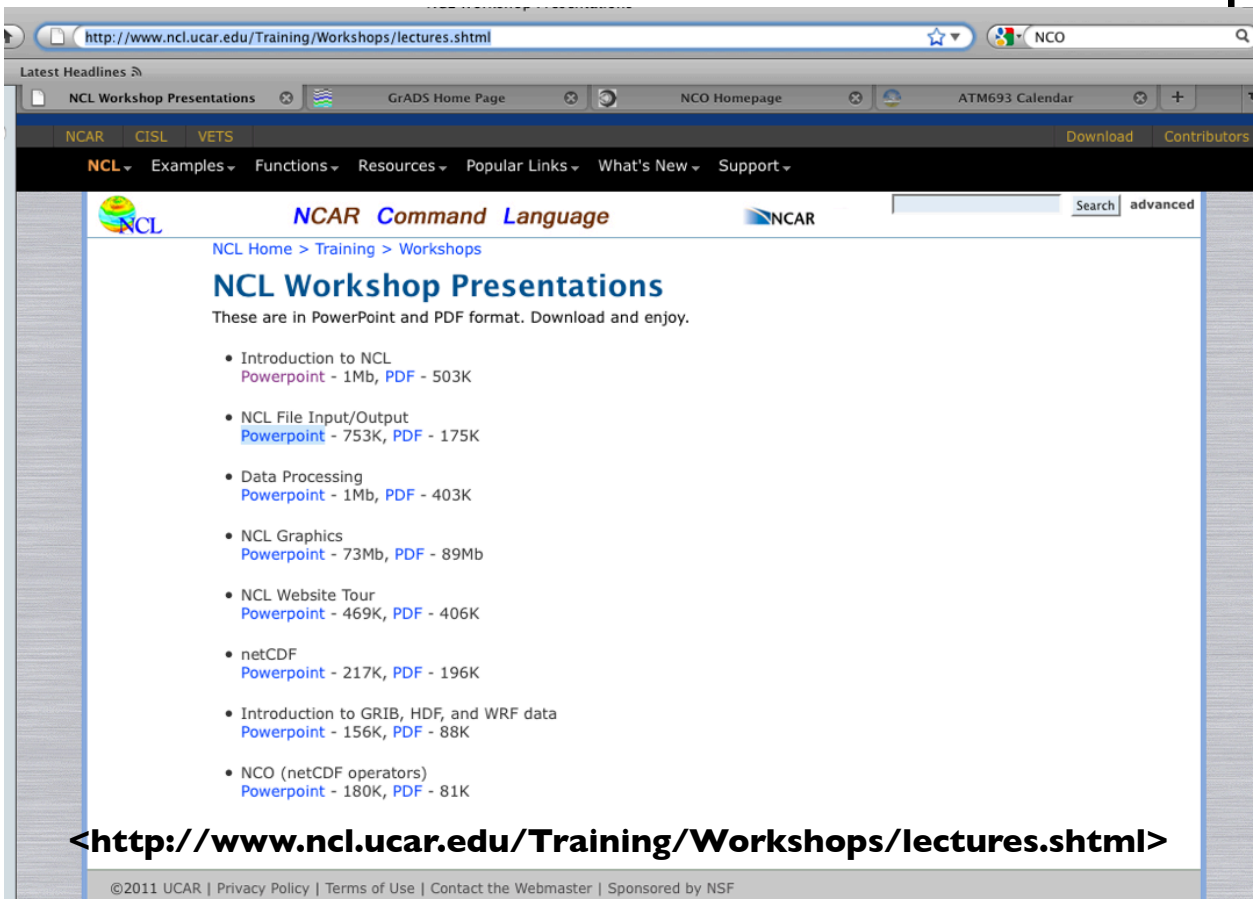
Class #3 Friday 28 January 2011

- 9:45-10:45, Back to this schedule next week. my travel
- What did we discuss last time?

Unix and VI

- Today (1.3.3 NCI Part 1)
- Will use NCL tutorial powerpoints, used in workshop held at UAF in 2003. Available online at NCL page



<http://www.ncl.ucar.edu/Training/Workshops/lectures.shtml>

Introduction Presentation

1. Reading in Netcdf data to NCL
2. Running NCL
3. Language Basics
4. How to define data structures
5. Printing information
6. I/O in NCL

How do you examine a netCDF file?

- **ncdump** file_name | less
  - dumps the entire contents of a file; prints every value
- **ncdump -h** file_name | less
  - Dumps header information [most commonly used]
  - NCL equivalent: **ncl_filedump file_name** | less
- **ncdump -v U** file_name | less
  - NCL equivalent: **ncl_filedump –v U file_name** | less
- Note: ncdump is a Unidata utility
  - **not** a netCDF Operator (NCO / CDO)
  - **not** associated with NCL

- **ncview:** visualize file contents [COARDS conven]

- **ncl_filedump** file_name [more general]
  - netCDF3/4, GRIB-1,GRIB-2, HDF, HDF-EOS [HDF5]

```
[uma-bhatts-mac-pro:Class_02/class02/Homework1_prob3] bhatt% ncdump -h sst.mnmean.nc
netcdf sst.mnmean {
dimensions:
        lon = 180 ;
        lat = 89 ;
        nbnds = 2 ;
        time = UNLIMITED ; // (1862 currently)
variables:
        float lat(lat) ;
                lat:units = "degrees_north" ;
                lat:long_name = "Latitude" ;
                lat:actual_range = 88.f, -88.f ;
                lat:standard_name = "latitude_north" ;
                lat:axis = "y" ;
                lat:coordinate_defines = "center" ;
        float lon(lon) ;
                lon:units = "degrees_east" ;
                lon:long_name = "Longitude" ;
                lon:actual_range = 0.f, 358.f ;
                lon:standard_name = "longitude_east" ;
                lon:axis = "x" ;
                lon:coordinate_defines = "center" ;
        double time(time) ;
                time:units = "days since 1800-1-1 00:00:00" ;
                time:long_name = "Time" ;
                time:actual_range = 19723., 76367. ;
                time:delta_t = "0000-01-00 00:00:00" ;
                time:avg_period = "0000-01-00 00:00:00" ;
                time:prev_avg_period = "0000-00-07 00:00:00" ;
                time:standard_name = "time" ;
                time:axis = "t" ;
        double time_bnds(time, nbnds) ;
                time_bnds:long_name = "Time Boundaries" ;
        short sst(time, lat, lon) ;
                sst:long_name = "Monthly Means of Sea Surface Temperature" ;
                sst:valid_range = -5.f, 40.f ;
                sst:actual_range = -1.8f, 34.24f ;
                sst:units = "degC" ;
                sst:add_offset = 0.f ;
                sst:scale_factor = 0.01f ;
                sst:missing_value = 32767s ;
                sst:precision = 2s ;
                sst:least_significant_digit = 1s ;
                sst:var_desc = "Sea Surface Temperature" ;
                sst:dataset = "NOAA Extended Reconstructed SST V3" ;
                sst:level_desc = "Surface" ;
                sst:statistic = "Mean" ;
                sst:parent_stat = "Mean" ;
```

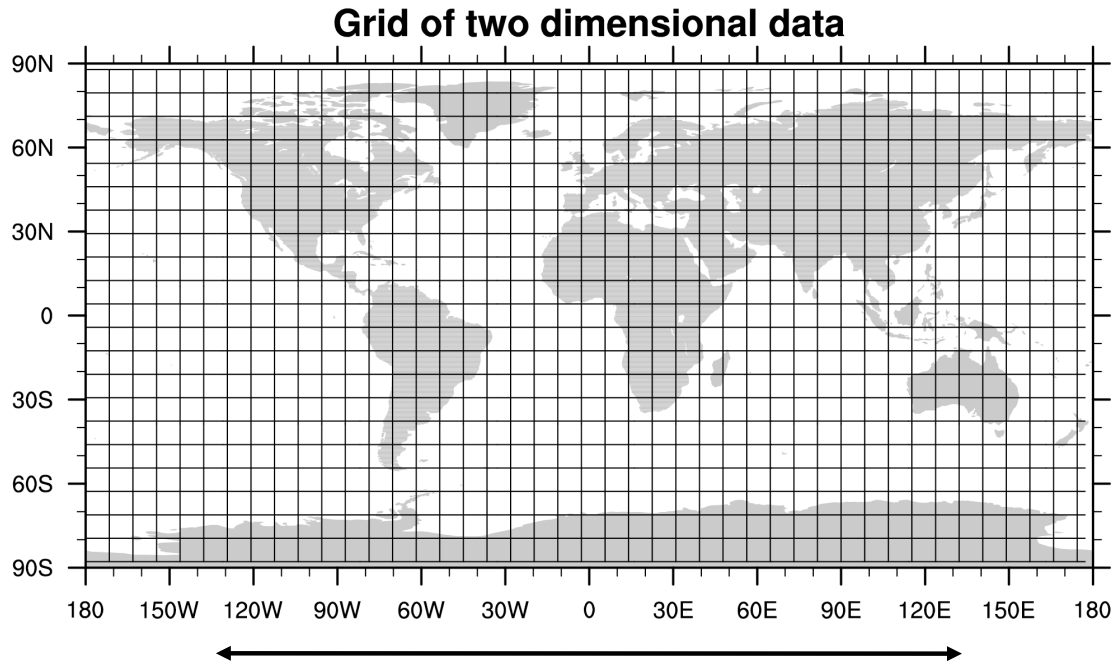
Look at our homework data file

Continue to look at the rest of our homework data file

```
// global attributes:
                :title = "NOAA Extended Reconstructed SST V3" ;
                :conventions = "CF-1.0" ;
                :history = "created 09/2007 by CAS" ;
                :comments = "The extended reconstructed sea surface temperature (ERSST)\n",
                        "was constructed using the most recently available \n",
                        "Comprehensive Ocean-Atmosphere Data Set (COADS) SST data \n",
                        "and improved statistical methods that allow stable \n",
                        "reconstruction using sparse data.\n",
                        "Currently, ERSST version 2 (ERSST.v2) and version 3 (ERSST.v3) are available from NCDC.\n",
                        " ERSST.v3 is an improved extended reconstruction over version 2.\n",
                        " Most of the improvements are justified by testing with simulated data.\n",
                        " The major differences are caused by the improved low-frequency (LF) tuning of ERSST.v3 \n",
                        "which reduces the SST anomaly damping before 1930 using the optimized parameters.\n",
                        " Beginning in 1985, ERSST.v3 is also improved by explicitly including\n",
                        " bias-adjusted satellite infrared data from AVHRR." ;
                :platform = "Model" ;
                :source = "NOAA/NESDIS/National Climatic Data Center" ;
                :institution = "NOAA/NESDIS/National Climatic Data Center" ;
                :references = "http://www.ncdc.noaa.gov/oa/climate/research/sst/ersstv3.php\n",
                        "http://www.cdc.noaa.gov/cdc/data.noaa.ersst.html" ;
                :citation = "Smith, T.M., R.W. Reynolds, Thomas C. Peterson, and Jay Lawrimore 2007: Improvements to NOAA\'s Historical Merged Land-Ocean Surf
ace Temperature Analysis (1880-2006). In press. Journal of Climate (ERSSTV3).\n",
                        "Smith, T.M., and R.W. Reynolds, 2003: Extended Reconstruction of Global Sea Surface Temperatures Based on COADS Data (1854-1997). Jou
rnal of Climate, 16, 1495-1510. ERSSTV1\n",
                        " Smith, T.M., and R.W. Reynolds, 2004: Improved Extended Reconstruction of SST (1854-1997). Journal of Climate, 17, 2466-2477.\000ERS
STv2" ;
```

visual: simple 2D netCDF Variable

coordinate variables (rectilinear grid)



attributes @:

- long_name
- units

NCL is **NOT LIMITED** to netCDF conforming variables

- eg: 2D coordinate arrays (curvilinear coords)

Open a netCDF in NCL

X

Scalar
or Array

attributes

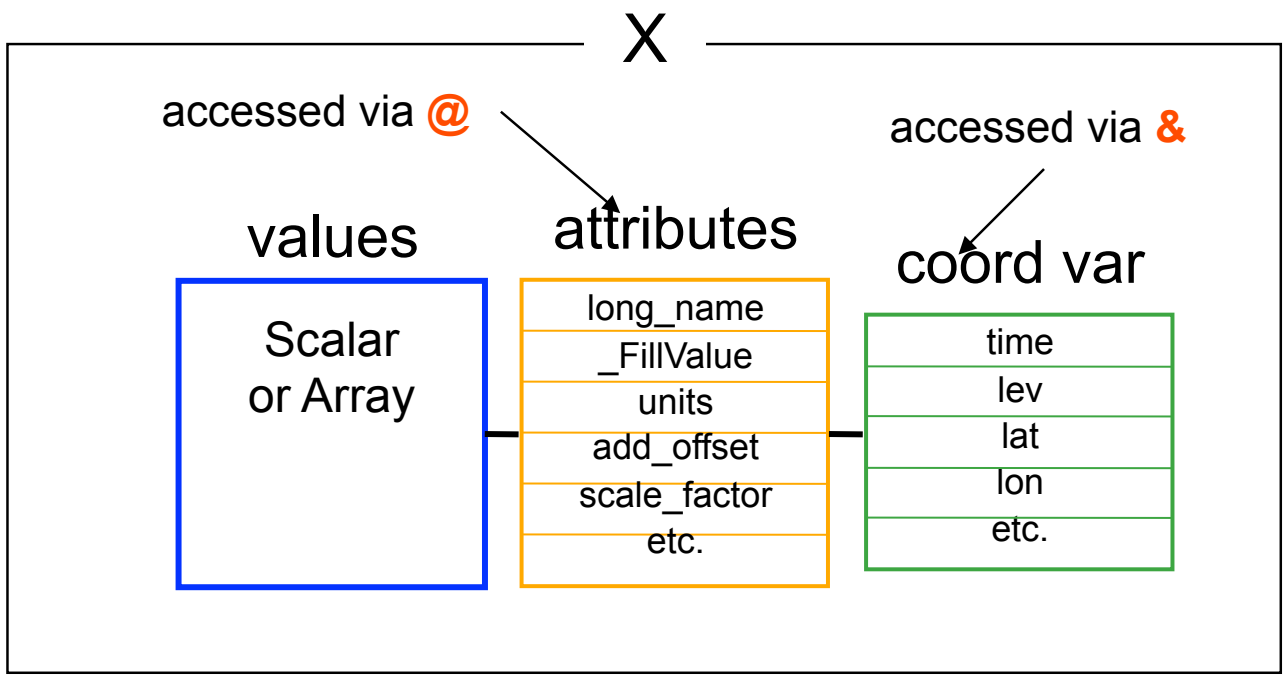
long_name
_FillValue
units
add_offset
scale_factor
etc.

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



Detailed Look netCDF Variable (NCL)

```
ncl <return>                                   ; interactive mode
ncl 0 > f = addfile ("UV300.nc", "r")          ; open file
ncl 1 > u = f->U                               ; import STRUCTURE
ncl 2 > printVarSummary (u)                    ; overview of variable
```

```
Variable:  u
Type: float
Total Size:  65536 bytes
             16384 values
Number of Dimensions:  3
Dimensions and Sizes:   [time | 2] x  [lat | 64] x [lon | 128]
Coordinates:
             time:  [ 1 .. 7 ]
             lat:  [ -87.8638 ..  87.8638 ]
             lon: [ 0 .. 357.185]
Number of Attributes: 5
    _FillValue :         1e36
    units :              m/s
    long_name  :         Zonal Wind Component
    short_name :         U
    missing_value :    1e36
```

Classic netCDF Variable Model

NCL syntax/funcs
query
use
modify
add
any aspect of data object

Running NCL

- **Interactive Mode (Command line)**
  - **ncl** *[options][command-line-arguments]* <return>
    ncl> enter commands
    ncl> **quit** <return>
  - can save interactive commands
    ncl> **record** "file_name"
    ncl> **stop record**

- **Batch Mode** [**.ncl** suffix is optional]
  - **ncl** *[options][arguments]* script.ncl
    - **ncl <** script.ncl [also acceptable]
  - **ncl** *[options][arguments]* script.ncl **>&!** out
  - **ncl** *[options][arguments]* script.ncl **>&!** out **&**
    - appending "**&**" means put in background
    - note: the **>&!** **&** are **csh** and **tcsh** syntax

Sample Batch Script: sample.ncl

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"

begin                                                    ; optional
 f1     = addfile("TMP_58-97.nc",  "r")                  ; open netCDF file
 T      = f1->Tmp                                        ; T(time,lev,lat,lon) =>(480,17,73,144)
 f2     = addfile("P_1958-1997.grb",  "r")               ; open GRIB file
 P      = f2->Pres                                       ; P(time,lev,lat,lon)
 f3     = addfile("Q_1958-1997.hdfeos",  "r")            ; open hdf –eos file
 Q      = f3->Specific_Humidity                          ; Q(time,lev,lat,lon)

 pot    = T* (1.+0.622*Q) *(1000/P)^0.286                ; potential temperature (array)
 print("pot:  min="+ min(pot) + "    max="+ max(pot) )

 wks  = gsn_open_wks("ps", "sample")                     ; open a graphic workstation
gsn_define_colormap (wks,"gui_default")                  ; change from default

 plot  = gsn_csm_contour_map_polar (wks,T(0,5,:,:),False)

 res   = True                                            ; change default plot
 res@cnFillOn               = True                       ; use colors
 res@gsnSpreadColors = True                              ; use entire color map
 plot = gsn_csm_pres_hgt  (wks,T(0,:,{50},:), res )
end                                                      ; only if begin is present
```

Outline: NCL Language Basics

- special syntax characters
- data types
- Variables → netCDF/NCL variable model
- attributes
- _FillValue
- named dimensions
- coordinate variables
- **print** and **printVarSummary**
- shaping
- subscripting

NCL Syntax Characters

- **;** - comment [can appear anywhere]
- **@** - reference/create attributes
- **!** - reference/create named dimension
- **&** - reference/create coordinate variable
- **{…}** - coordinate subscripting
- **$** - enclose strings when (im/ex)port variables via addfile
- **(/../)** - array construct characters
- **:** - array syntax
- **|** - separator for named dimensions
- **** - continue character [statement to span multiple lines]
- **::** - syntax for external shared objects (eg, fortran/C)
- **->** - use to (im/ex)port variables via **addfile** function

Arrays

- **row major** ….. like C/C++
  - left dimension varies slowest; right varies fastest
  - dimension numbering left to right [0,1,..]
- indicies [subscripts] are zero based [0,N-1]

Consider T(:,:,:)

left dimension is 0 [T!0]
middle dimension is 1 [T!1]
right dimension is 2 [T!2]

NCL (netCDF): Named Dimensions

- **may be "named"**
  - provides alternative way to reference subscripts
  - recommendation: always name dimensions
  - use NCL syntax

- **assigned with ! character** {let T(:,:,:)}
  - 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 = "LAT" … dName = T!2
- **delete** can eliminate: delete (T!2)

- **Named dimensions used to reshape**

Create and Assign Coordinate Variables

- **create 1D array**
  - time = (/ 1980, 1981, 1982 /) ; integer
  - lon = ispan(0, 355, 5)*1.0 ; integer->float

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

- **assign values to named dimension**
  - time&time = time
  - lon&lon = lon

- **let x be 2D: name dimensions**
  - x!0 = “time” … x!1 = “lon”
- **assign coordinate variables to x**
  - x&time = time … x&lon = lon

Variable Assignment

- **Variable-to-Variable assignment**
  - consider **y = x** where **x** is previously defined
    - if **y** not defined:
      - **y** has same type/shape/values/meta data as x
    - if **y** predefined:
      - **y** must have same shape and type
      - or, **x** must be coerceible to the type of y
      - **y** attributes, dimension names and coordinate variables, if they exist, will be over written

- **Value-only assignment (no meta copy)**
  - U multi-dimensional array with meta data
    - Uval = (/ U /) or Uval = (/ f->U/)
    - the (/ ... /) operator pair strips meta data

Variable Subscripting (1 of 3)

Standard Array Subscripting

- ranges: start/end and [optional] stride
- indices separated by :
- 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
(0,::-1,:50)	➔	1st time index, reverse lat order, 1st 51 lon
T(:1,45,10:20)	➔	1st 2 time indices, 46th index of lat, 10-20 indicies of lon

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

“printing”

- **printVarSummary**
  - gives gross overview of a variable

- **print**
  - same info as printVarSummary
  - prints values

- **write_matrix**
  - print to standard out or a file
  - format control of numerical output
  - can write to file also

printVarSummary

- **Print out variable (data object) information**
  - type
  - dimension information
  - coordinate information (if present)
  - attributes (if present)
- **printVarSummary** (u)

```
Variable: u
Type: double
Total Size: 1179648 bytes
            147456   values
Number of Dimensions:  4
            Dimensions / Sizes:  [time | 1]  x  [lev | 18]  x  [lat | 64] x  [lon | 128]
Coordinates:
            time:   [4046..4046]
            lev:    [4.809 .. 992.5282]
            lat:    [-87.86379 .. 87.86379]
            lon:    [ 0. 0 .. 357.1875]
Number of Attributes:  2
            long_name: zonal wind component
            units:          m/s
```

print (1 of 3)

- **Prints out all variable information including**
  - meta data, values
  - T(lat,lon): print (T)

```
Variable: T
Type: float
Total Size: 32768 bytes
            8192   values
Number of Dimensions:  2
            Dimensions / Sizes: [lat | 64] x  [lon | 128]
Coordinates:
           lat:      [-87.86379 .. 87.86379]
            lon:     [ 0. 0 .. 357.1875]
Number of Attributes:  2
            long_name: Temperature
            units:           C
(0,0)  -31.7
(0,1)  -31.4
(0,2)  -32.3
(0,3)  -33.4
(0,4)  -31.3    etc. [entire T array will be printed]
```

print (2 of 3)

- **can be used to print a subset of array**
  - meta data, values
  - T(lat,lon): print(T(:,103)) or print(T(:,{110}))

```
Variable: T (subsection)
Type: float
Total Size: 256 bytes
            64   values
Number of Dimensions:  1
            Dimensions / Sizes: [lat | 64]
Coordinates:
          lat:      [-87.86379 .. 87.86379]
Number of Attributes:  3
            long_name: Temperature
            units:          C
            lon:            109.6875    [ added ]
(0)  -40.7
(1)  -33.0
(2)-25.1
(3)-20.0
(4)  -15.3    etc.
```

print (3 of 3)

- **print with embedded strings**
  - no meta data
  - print ("min(T)="+min(T)+" max(T)="+max(T))

```
(0)    min(T)=-53.8125   max(T)=25.9736
```

- **sprintf and sprinti provide formatting**

  - often used in graphics
  - print ("min(T) = "+ sprintf("%5.2f ", min(T)))

```
(0)    min(T) = -53.81
```

- **sprinti can left fill with zeros (ex: let n=3)**

  - fnam = "h" + sprinti ("%0.5i", n) + ".nc"
  - print("file name = "+fnam)

```
(0)    file name = h00003.nc
```

write_matrix(x[*][*], fmt, opt)

- **pretty-print 2D array (table) to standard out**
  - integer, float, double
  - user format control (fmt)
  - T(N,M), N=7, M=5: **write_matrix** (T, “5f7.2”, False)

```
 4.35    4.39    0.27   -3.35   -6.90
 4.36    4.66    3.77   -1.66    4.06
 9.73   -5.84    0.89    8.46   10.39
 4.91    4.59   -3.09    7.55    4.56
  17    3.68    5.08    0.14   -5.63
-0.63   -4.12   -2.51    1.76   -1.43
-4.29    0.07    5.85    0.87    8.65
```

- **can also create an ASCII file**

```
opt          = True
opt@fout = “foo.ascii”                ; file name
write_matrix (T, “5f7.2”, opt)
```

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**(f, "DefineMode" ,True)
  - reading GRIB
    - **setfileoption**("grb" ,"ThinnedGridInterpolation", "cubic")
    - **setfileoption**("grb", "InitialTimeCoordinateType" \
      , "Numeric")
  - reading/writing Binary
    - **setfileoption**("bin", "ReadByteOrder", "LittleEndian")
    - **setfileoption**("bin", “WriteByteOrder", “BigEndian")

addfile (1 of 3)

- Used to open a **supported** format only

- **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"**, **"he5"** [read/write]
    - GRIB: **"grb"** , **"grib"** [read only; GRIB1 or GRIB2]
    - CCMHT: **"ccm"** [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**")

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

```
diri  = "/fs/cgd/data0/shea/ccm/"
fili  = "testCCM"
ext   = ".ccm"
fin   = addfile(diri+fili+ext , " 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
```

Example: open, read, output netCDF

```
begin        ; optional
;---------------------------------------
;open file and read in data
;---------------------------------------
  fin    = addfile ("in.nc, "r")
  u      = fin->U
;---------------------------------------
;create reference to output file
;---------------------------------------
   fout   = addfile("out.nc" , "c")
;---------------------------------------
;add a global attribute to the file
;---------------------------------------
   fout@title = "I/O Example 1"
;---------------------------------------
;Output variable u to netCDF file
;---------------------------------------
   fout->U= u
end          ; only if begin is present
```

Note: this method of outputing a netCDF file has simple syntax, but can be slow

Reading Binary/ASCII data

- **7 functions for reading binary:**
  - **fbinrecread**: reads multiple unformatted sequential records [Fortran; ieee]
  - **fbinnumrec**: returns the number of unformatted sequential records [Fortran; ieee]
  - **fbindirread**: reads specified record from a Fortran direct access file [ieee]
  - **fbinread**: same as **fbinrecread** but reads only one ieee rec
  - **craybinrecread**: like fbinrecread but for COS blocked data
  - **craybinnumrec**: like fbinnumrec but for COS blocked data
  - **cbinread**: read binary created via C block IO function "write"

- **1 function for reading ASCII data:**
  - **asciiread** **[contributed.ncl: readAsciiTable]**
  - use Fortran/C to read complicated ASCII files

- **all above functions allow data to be shaped**
  - x = **fbinrecread** ("foo_ieee", rnum, (/10,20,30/), "float")
  - a = **asciiread** ("foo_ascii", (/64,128/) , "float")

Writing Binary/ASCII data

- **4 procedures for writing (ieee) binary data**
  - **fbinrecwrite**: write unformatted fortran sequential recs
  - **fbindirwrite**: write specified record; fortran direct access
  - **fbinwrite**: write a binary file containing a single record
  - **cbinwrite**: write binary file ; mimics C block IO "write"
- **setfileoption:** can be used to alter behavior

- **2 procedures to write ascii data**
  - **asciiwrite**: write a file containing ASCII characters
    - writes a single flat ASCII file. One value per line.
    - No user control of format
  - **write_matrix**: write a multi-dim array to std out or to a file
    - user has format control … pretty-print
    - options for title and row numbering

- use Fortran/C to write complicated ASCII files.

netCDF,GRIB,HDF ==> binary

```
  fin    = addfile ("in.nc", "r")     ; .grb   .hdf   hdfeos
  u      = fin->U
  v      = fin->V
  t      = fin->T
  fout   = "out.bin"
  system ("/bin/rm –f  "+fout)
;---------------------------------------------------
; output binary: –1 means append to previous rec
;---------------------------------------------------
 setfileoption("bin", “WriteByteOrder", “BigEndian")

  fbinrecwrite (fout, -1, fin->time)
  fbinrecwrite (fout, -1, fin->lev)
  fbinrecwrite (fout, -1, fin->lat)
  fbinrecwrite (fout, -1, fni->lon)
  fbinrecwrite (fout, -1, u)
  fbinrecwrite (fout, -1, v)
  fbinrecwrite (fout, -1, t)
```

binary ==> netCDF

```
; read in data
    lat    = fbinrecread ("./in.bin", 2,  64, "double")
    lon    = fbinrecread ("./in.bin", 3,128, "double")
    u      = fbinrecread ("./in.bin", 6, (/64,128/),"double")

    lat!0                  = "lat"
    lat@long_name    = "latitude"
    lat@units              = "degrees_north"
    lon!0                  = "lon"
    lon@long_name   = "longitude"
    lon@units               = "degrees_east"

    u!0       = "lat“                          ; named dimensions
    u!1       = "lon"
    u&lat     =  lat                           ; coordinate variables
    u&lon     =  lon
    u@long_name = "zonal wind“  ; attributes
    u@units           = "m/s"

    fout          = addfile ("out.nc", "c")   ; output file
    fout@title = "Binary-to-netCDF"      ; file attribute
    fout->U     = u                              ; write variable to file
```