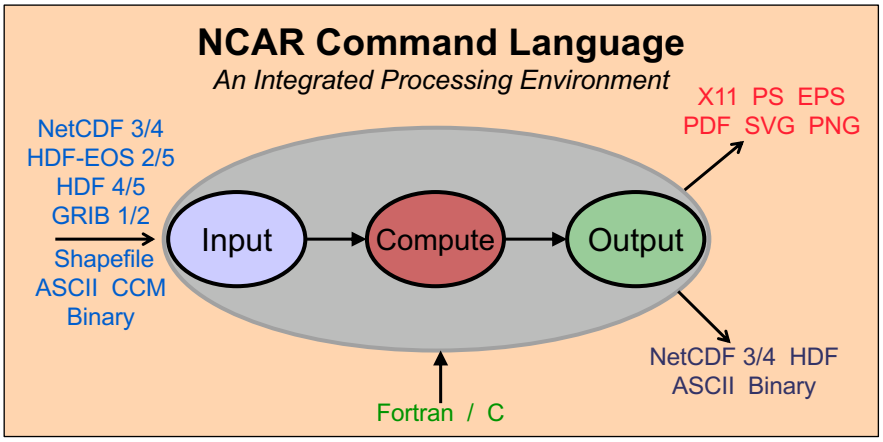
Introduction to NCL File I/O



Mary Haley
(with thanks to Dennis Shea)

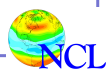
Sponsored by the National Science Foundation

NCL File I/O Outline

- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusions

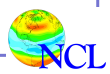
NCL File I/O Outline

- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



Goals

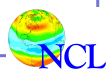
- Give overview of data formats and tools
- Intersperse demos, tips, and useful links
- **Help you understand your data**



Two takeaways

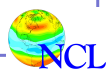
1. LOOK AT YOUR DATA

2. KNOW YOUR DATA



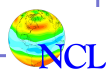
Why is this important?

- Can't make assumptions that data is correct
- Does the data need to be unpacked?
- Know the units of your data
- Match tools with data
- Understand the coordinates of your data (time, lat, lon, etc)
- If data is large, may need to watch memory usage



NCL File I/O Outline

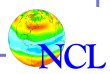
- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



Data formats: self-describing

Self-describing data formats are files that contain data values, plus descriptive information about the values ("metadata").

Metadata is information about the file itself and about the variables on the file



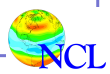
Data formats: self-describing

Metadata generally includes:

- Attributes
  - Descriptive information about file

    creation_date, *conventions*, *history*, *source*, *revision_id*
  - Descriptive information about variables

    long_name, *units*, *_FillValue*, *valid_range*, *add_offset*, *scale_factor*
- Dimension names and sizes
- Coordinate information
  - *time*
  - *level*
  - *latitude* / *longitude* (*lat* / *lon*)



NCL's supported file formats

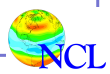
- NetCDF3 / NetCDF4

  [Network Common Data Form]
- HDF4 / HDF5

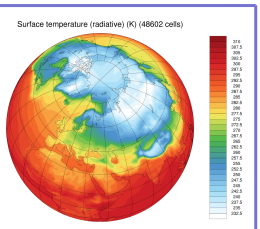
  [Hierarchical Data Format]
- HDF-EOS 2 / HDF-EOS 5

  [Earth Observing System]
- GRIB1 / GRIB2

  [Gridded Binary, WMO standard, NCEP, ECMWF]
- Shapefile



NetCDF



- Most common in climate sciences
- Developed and supported by Unidata
- Two versions: NetCDF-3 and NetCDF-4

  http://www.unidata.ucar.edu/software/netcdf/
- Conventions

  http://www.unidata.ucar.edu/software/netcdf/conventions.html

NetCDF Conventions

- CF (Climate and Forecast) Conventions
  http://cfconventions.org/
- More conventions
  http://www.unidata.ucar.edu/software/netcdf/conventions.html

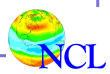
Look for "Conventions" attribute on file

Check your NetCDF file

Tip

CF-convention compliance checker

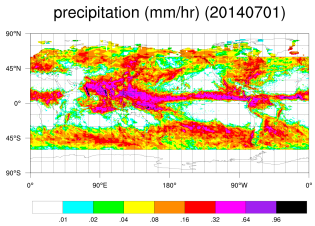
http://puma.nerc.ac.uk/cgi-bin/cf-checker.pl

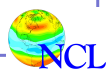


HDF

- Tailored for large and complex datasets
- Used by a wide variety of scientific disciplines
- Two versions HDF4 / HDF5

http://www.hdfgroup.org

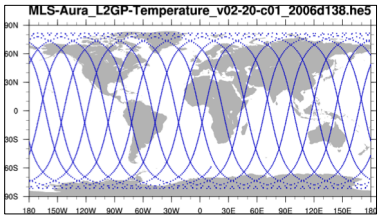




HDF-EOS

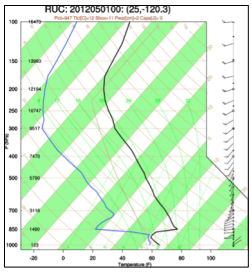
- HDF4 and HDF5 subset with conventions, data types, and metadata
- Used for NASA EOS missions (mostly satellite)
- Geo-located data

http://hdfeos.net



GRIB

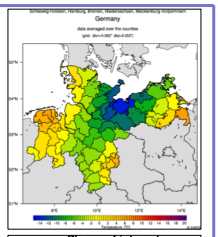
- World Meteorological Organization standard
- Historical / forecast weather data
- Actually a "record" format
- Requires look-up tables for the metadata (GRIB code tables)
- NCL has 50+ built-in tables
- Latitude / longitude arrays created for you

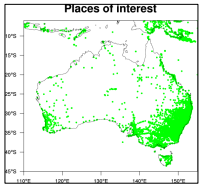


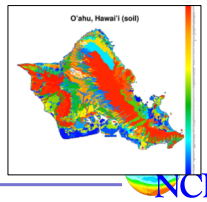
General Regularly-distributed Information in Binary form

Shapefile

- ESRI/GIS format
- Can be points, lines, polygons
- Example: population, roads, country boundaries, election data, airport locations
- Used for graphics, masking data







One function to read them all

addfile

Users need not "fear" any supported format (generally…)

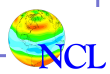
User need not know internal structure of supported files

NCL imports variables from all supported files into a common data structure

ASCII files

- Quick and easy to look at
- CSV (comma-separated values) are ASCII files
- No conventions!
- Can be quite large and unwieldy
- NCL has many functions for handling ASCII

http://www.ncl.ucar.edu/Applications/list_io.shtml



Sample CSV (comma-separated values) file

IBTrACS WMO: International Best Tracks Archive for Climate Stewardship -- WMO DATA ONLY -- Version: v03r04
Serial_Num,Season,Num,Basin,Sub_basin,Name,ISO_time,Nature,Latitude,Longitude, Wind(WMO),Pres(WMO),Center,Wind(WMO) Percentile,Pres(WMO) Percentile,Track_type
N/A,Year,#,BB,BB,N/A,YYYY-MM-DD HH:MM:SS,N/A,deg_north,deg_east,kt,mb,N/A,%,%,N/A
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-11 06:00:00, NR, -8.60, 79.80, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-12 06:00:00, NR, -9.00, 78.90, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-13 06:00:00, NR,-10.40, 73.20, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-14 06:00:00, NR,-12.80, 69.90, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-15 06:00:00, NR,-13.90, 68.90, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-16 06:00:00, NR,-15.30, 67.70, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-17 06:00:00, NR,-16.50, 67.00, 0.0, 0.0,reunion,-100.000,-100.000,main
1848011S09080,1848,02, SI, MM,XXXX848003,1848-01-18 06:00:00, NR,-18.00,

Sample (nightmarish) ASCII file

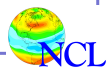
USC00040029189403TMAX-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 6 6 28
6 39 6 56 6 117 6 150 6 172 6 156 6 167 6 178 6 117 6 128 6 111 6
USC00040029189403TMIN-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -83 6 -67
6 -44 6 -78 6 -50 6 -39 6 -11 6 11 6 39 6 50 6 56 6 -11 6 0 6
USC00040029189403PRCP 76 6 0P 6 0P 6 0P 6 51 6 0P 6 0P 6 0P 6
25 6 0P 6 0P 6 0P 6 0P 6 0P 6 20 6 0P 6 0P 6 0P 6 18 6 0P 6 0P
6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 81 6 53 6 0P 6
USC00040029189403SNOW 76 6 0 6 0 6 0 6 51 6 0 6 0 6 0 6 25 6
0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6
0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 6
USC00040029189404TMAX 144 6 106 6 128 6 167 6 172 6 183 6 139 6 161
6 217 6 194 6 117 6 161 6 178 6 139 6 56 6 94 6 178 6 194 6 233 6 217
6 211 6 144 6 117 6 189 6 167 6 56 6 39 6 94 6 167 6 183 6-9999
USC00040029189404TMIN 28 6 -11 6 -6 6 -6 6 -6 6 6 6 61 6 -6 6 28 6
50 6 6 6 -22 6 11 6 22 6 -11 6 -39 6 -33 6 0 6 22 6 67 6 61 6 61 6
22 6 -6 6 61 6 -17 6 -50 6 -11 6 33 6 17 6-9999
USC00040029189404PRCP 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6
0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P 6 0P
6 0P 6 0P 6 0P 6 0P 6 211 6-9999 0P 6 0P 6 0P 6-9999

Binary files

- Usually written by a C or Fortran program
- Cannot look at the file with a UNIX editor
- Hard to read if you don't know how it was written
- NCL has functions for reading/writing binary

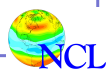
```
fbinread, fbindirread, fbinrecread,
fbinnumrec, fbinwrite, fbinrecwrite,
cbinread, cbinwrite
```

http://www.ncl.ucar.edu/Applications/list_io.shtml



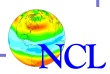
NCL File I/O Outline

- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



NCL and other useful tools

- Part of NCL distribution
  - ncl_filedump
  - ncl_convert2nc
- PyNIO (Python package based on NCL's file I/O)
- NetCDF
  - ncdump
- NetCDF Operators (NCO)
  - ncks, ncrcat, etc
- Climate Data Operators (CDO)
- Quick viewers
  - ncview
  - panoply



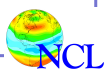
NCL's ncl_filedump

- Run it from the UNIX command line
- Similar to "ncdump –h"
- For usage and list of options:

```
ncl_filedump  -h
```

- Provides textual overview of any supported file's contents
- File name doesn't need suffix, but must provide one to ncl_filedump

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

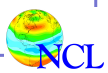


Dumping contents of supported files

Tip

NetCDF:	ncdump
HDF4:	hdp
HDF5:	h5dump
GRIB1:	wgrib
GRIB2:	wgrib2

These tools have to be installed separately; they are not part of NCL



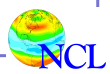
Tip

NetCDF tip

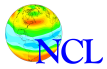
Determining the type of your NetCDF file

```
ncdump -k  my_netcdf_file
```

classic 64-bit offset netCDF-4
netCDF-4 classic



Demo #1
ncl_filedump



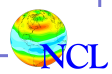
NCL's ncl_convert2nc

- Convert supported file to NetCDF
- Run it from the UNIX command line
- For usage and list of options:

```
ncl_convert2nc -h
```

- Special options for handling GRIB time
- Some HDF5 files cannot be converted due to complexity (group, compound)

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



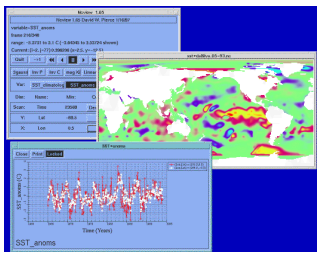
Tip Quick view of supported files

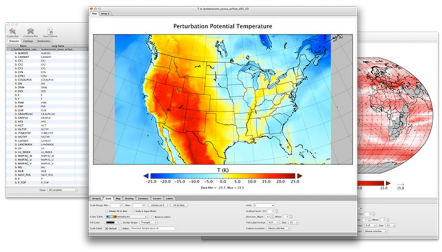
ncview – NetCDF visual browser

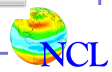
http://meteora.ucsd.edu/~pierce/ncview_home_page.html

panoply – NetCDF, GRIB, HDF viewer

https://www.giss.nasa.gov/tools/panoply/





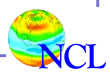


Doing quick operations across multiple NetCDF files

Tip

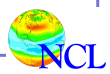
NetCDF Climate Operators (NCO)
http://nco.sourceforge.net/

Climate Data Operators (CDO)
https://code.zmaw.de/projects/cdo



NCL File I/O Outline

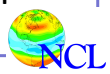
- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



addfile – opens supported file (1 of 4)

- One of the most powerful functions in NCL
- Opens a "supported" file for reading or writing:
  - read ("r")
  - create ("c")
  - read/write ("w")
- Variables read off file will "look like NetCDF"
- Opens file based on extension

  ".nc", ".cdf", ".nc4", ".grb", ".grb1", ".grb2", ".hdf4", ".hdf5", ".he2", ".he5", ".hdfeos", ".shp"
- Actual file name not required to have extension



addfile – looking at your file (2 of 4)

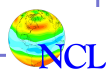
Equivalent to "ncdump –h" or "ncl_filedump"

```
f = addfile("b30.061cb_ANN_globalClimo.nc", "r")
print(f)
```

```
f = addfile("../MLS-Aura_L2GP-IWC_v02.he5", "r")
print(f)
```

```
fin = addfile("/dss/dsxxx/Y12345.grb", "r")
print(fin)
```

```
dir   = "/home/haley/wrf_files/"
fname = "wrfout_d01_2003-07-15_00:00:00"
wfile = addfile(dir + fname + ".nc", "r")
print(wfile)
```

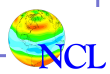



addfile – OPeNDAP enabled (3 of 4)

Open Source Project for Network Data Access Protocol

- Allows remote access of file over the internet
- File must be on an OPeNDAP server
- Some OPeNDAP servers require registrations/logons
- Works with isfilepresent, addfile and addfiles

```
;---print contents of OPeNDAP file served by NOAA
url="http://www.esrl.noaa.gov/psd/thredds/dodsC/Datasets/ncep.reanalysis.dailyavgs/pressure/"
fname = "air.1948.nc"
f     = addfile(url + fname, "r")
print(f)
```

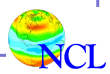



addfile – query functions (4 of 4)

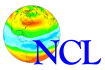
Swapped slides

- getfilevarnames
- getfilevaratts
- getfilevardimsizes
- getfilevartypes
- isfilepresent
- isfilevaratt
- isfilevardim
- isfilevarcoord

```
;---Print all the variables on a supported file
f = addfile("b30.061cb_ANN_globalClimo.nc", "r")
print(getfilevarnames(f))

;---Check if a variable is on the file
f = addfile("sst8292.nc", "r")
print(isfilevar(f,"SST"))     ; True
print(isfilevar(f,"temp"))    ; False
```

Demo #2
addfile



PyNIO – looking at your file

Also equivalent to "ncdump –h" or "ncl_filedump"

```
from Nio import open_file
fname = "MET9_IR108_cosmode_0909210000.grb2"
f     = open_file(fname)
print(f)
```

```
Nio file : MET9_IR108_cosmode_0909210000.grb2
  dimensions:
     ygrid_0 = 461
     xgrid_0 = 421
  variables:
     float SBTMP_P31_GRLL0 [ ygrid_0, xgrid_0 ]
        center :Offenbach (RSMC)
        production_status :Operational products
        long_name :Scaled brightness temperature
        units :numeric
        _FillValue :1e+20
        coordinates :gridlat_0 gridlon_0
        grid_type :Rotated latitude/longitude
        forecast_time :0
        forecast_time_units :hours
        initial_time :09/21/2009 (00:00)
```

Reading a variable

- Use -> syntax:

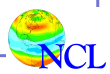
```
f = addfile("atmos.nc","r")
t = f->T     ; Reads var and metadata
```

- Can subset the variable when you read it:

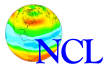
```
f = addfile("atmos.nc","r")
t = f->T(0,:,:)    ; 3D var to 2D var
```

- LOOK AT YOUR DATA!

```
printVarSummary(t)
printMinMax(t,0)
```

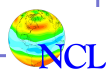



Demo #3
Reading a variable



Looking at your data

- printVarSummary – look at the metadata!
  - What size is the array?
  - What type is it?
  - What are the long_name and units?
  - Does it have coordinate variables?
  - Does it have a "coordinates" attribute?
  - Does it have a _FillValue or missing_value?
  - Does it have add_offset and scale_factor attributes?
  - Is the array large?
- printMinMax – do the values look correct?



Reading a variable with non-standard characters

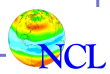
Tip

- Variable name has a dash: "cbbr-msk"
- Use ***$"varname"$*** syntax:

```
f = addfile("masks_tbbr.nc", "r")
amask = f->cbbr-msk(0,:)      ; NO!
amask = f->$"cbbr-msk"$(0,:)  ; YES!
```

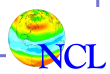
- Can assign variable name to a variable:

```
f = addfile("masks_tbbr.nc", "r")
vname = "cbbr-msk"   ; Don't forget double quotes
amask = f->$vname$(0,:)
```

Reading a variable with PyNIO

```
import Nio as nio
import numpy as np

#---Open file
filename = "MOD06_L2.A2010031.1430.005.2010031221343.hdf"
f = nio.open_file(filename)

#---Print file metadata
print(f)

#---Read "Cloud_Top_Temperature"
ctt = f.variables["Cloud_Top_Temperature"] # NioVariable

#---Print information about ctt
print("Shape is",ctt.shape)
print("Type is",type(ctt))
print("min/max CTT = %g / %g" % (np.min(ctt[:]),np.max(ctt[:])))
```

Issues to watch for (1 of 3)

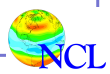
- Do the min/max values look correct?
- If not, check the units and _FillValue attribute.
- Does variable have add_offset / scale_factor type attributes?

  If so, may need to be unpacked (*):

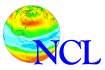
```
u = short2flt(f->U) ; if U is a short
u = byte2flt(f->U)  ; if U is a byte

;---This is what happens "under the hood"
us = f->U
u  = us*us@scale_factor + us@add_offset
```

* Equation for unpacking is NOT standard. Attribute names could be different. Make sure you know the correct equation to use.

```
w = wv1s@scale_factor * (wv1s - wv1s@add_offset)
```

Demo #4
Unpacking a variable



Issues to watch for (2 of 3)

- Is the value 0.0 being used as a missing value? If so, need to set it to something else. Two examples:

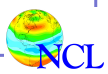
```
x@_FillValue = 1e20
x@_FillValue = default_fillvalue(typeof(x))
```

- Does the data have a _FillValue and a missing_value attribute that aren't the same?

```
x@_FillValue = x@missing_value  ; Easy fix!
```

- Does the data have NaNs? (Not-A-Number)

```
if(any(isnan_ieee(x))) then
  replace_ieeenan (x,x@_FillValue,0)
end if
```

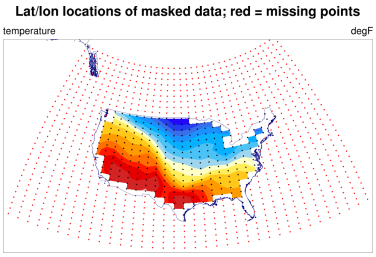



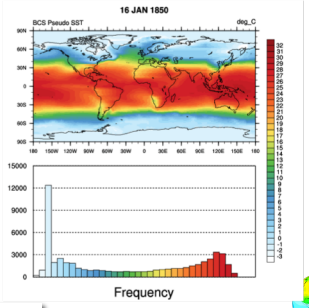
Issues to watch for (3 of 3)

- Is the data sparse? That is, do you have a lot of missing values?

```
num_valid = num(.not.ismissing(x))
num_msg   = num(ismissing(x))
print("x has " + num_valid + " valid values")
print("x has " + num_msg + " missing values")
```

- How is the data distributed? Use plotting techniques to check.





NCL

Demo #5
Fixing a variable with "bad" missing value attribute

NCL

setfileoption procedure

- NCL procedure for customizing behavior of file I/O functions
- Allows you to set specify file-format-specific options

Writing netCDF

```
setfileoption(f, "DefineMode", True)
setfileoption("nc", "Format", "LargeFile")
setfileoption("nc", "Format", "netCDF4Classic")
```

Reading GRIB

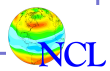
```
setfileoption("grb", "ThinnedGridInterpolation",  "cubic")
setfileoption("grb", "InitialTimeCoordinateType", "Numeric")
```

Reading/writing binary

```
setfileoption("bin", "ReadByteOrder",  "LittleEndian")
setfileoption("bin", "WriteByteOrder", "BigEndian")
```


addfiles - Open multiple supported files

- Similar to addfile, but there are differences
- addfile returns a file pointer, addfiles returns a list of file pointers
- Can open a series of files in two modes:
  1. "cat" (default, leftmost dimension is concatenated)
  2. "join" (new leftmost dimension added)
- Use ListSetType procedure to change modes



addfiles – "cat" versus "join"

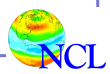
```
fnames = systemfunc("ls pottmp.*.nc") ; 29 files, each
fall = addfiles (fnames, "r")         ; with 12 timesteps

;---Read "pottmp" in default "cat" mode
pottmp = fall[:]->pottmp    ; note syntax [:]
printVarSummary(pottmp)     ; LOOK AT YOUR DATA!

; [time | 348] x [level | 40] x [lat | 418] x [lon | 360]
```

```
fnames = systemfunc("ls pottmp.*.nc")
fall = addfiles (fnames, "r")
ListSetType (fall, "join")

;---Read "pottmp" in "join" mode
pottmp = fall[:]->pottmp    ; note syntax [:]
printVarSummary(pottmp)

;  [ncl_join | 29] x [time | 12] x [level | 40] x [lat | 418] x [lon | 360]
```

Demo #6
addfiles

NCL File I/O Outline

- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



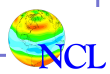
addfile – Creating a NetCDF file

Assume you already have a variables "u" and "Temp" that you want to write to a file

```
fout = addfile ("newfile.nc", "c")
fout->U = u      ; All of u's and Temp's
fout->T = Temp   ; metadata will be written
```

Note: this is considered an INEFFICIENT method for writing NetCDF.
It's fine for small variables, or a quick test.

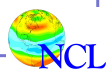
For large files or lots of variables, EFFICIENT method is recommended.



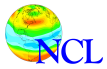
Slightly more advanced example

```
fname = "foo.nc"
;---Removes file if it exists
system("rm " + fname)

;---Open file in "create" mode
fout = addfile (fname, "c")
;---Add file attributes
fout@title  = "Simple Example"
fout@creation_date = systemfunc("date")

;---Good idea to make time unlimited
filedimdef (fout, "time", -1, True)
fout->U  = u
fout->T  = Temp
```

Note: this is still considered an inefficient method of writing NetCDF.

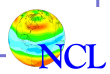


Demo #7
Writing a NetCDF file the easy way



Two ways to create a NetCDF file

1. Inefficient, but easy method
2. Efficient, but more tedious – a MUST for large files and/or lots of variables
   - Define file attributes (optional , recommended)
   - Define dimension names and sizes
   - Define variable names, sizes, types
   - Define variable attributes (optional, recommended)
   - Write variable values to file, using (/…/) syntax



Efficient NetCDF creation

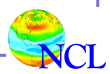
Requires use of following procedures:

- setfileoption – enter define mode
- fileattdef – define file attributes
- filedimdef – define dimensions
- filevardef – define variables
- filevarattdef – define variable attributes

```
Timing comparison:     500 3D variables, 10 x 180 x 360
Inefficient method:    168 secs
Efficient method:        9 secs
```

Examples that show the two methods:

http://www.ncl.ucar.edu/Applications/write_netcdf.shtml



Efficient method

```
nlat  = dimsizes(lat)
nlon  = dimsizes(lon)

fout = addfile ("newfile.nc", "c")
setfileoption(fout,"DefineMode",True)

fAtt               = True
fAtt@title         = "Efficient NetCDF"
fAtt@Conventions   = "None"
fAtt@creation_date = systemfunc ("date")
fileattdef( fout, fAtt )

dimNames = (/"time", "lat", "lon"/)
dimSizes = (/ -1   ,  nlat,  nlon /)
dimUnlim = (/ True , False, False/)
filedimdef(fout,dimNames,dimSizes,dimUnlim)

filevardef(fout, "time" ,typeof(time),getvardims(time))
filevardef(fout, "lat"  ,typeof(lat),getvardims(lat))
filevardef(fout, "lon"  ,typeof(lon),getvardims(lon))
filevardef(fout, "T"    ,typeof(T)  ,getvardims(T))
filevardef(fout, "TOPOG",typeof(ORO),getvardims(ORO))

filevarattdef(fout,"T",T)
filevarattdef(fout,"time" ,time)
filevarattdef(fout,"lat"  ,lat)
filevarattdef(fout,"lon"  ,lon)
filevarattdef(fout,"TOPOG",ORO)

fout->time   = (/time/)
fout->lat    = (/lat/)
fout->lon    = (/lon/)
fout->T      = (/T/)
fout->TOPOG  = (/ORO/)
```

Assume T and ORO have coordinate arrays "time", "lat", and "lon"

Step 1: Define global attributes

Step 2: Define dimensions

Step 3: Define variables

Step 5: Define variable attributes

```
fout = addfile ("newfile.nc", "c")
fout@title = "Inefficient NetCDF"
fout@Conventions = "None"
fout@creation_date = systemfunc ("date")

filedimdef(fout,"time",-1,True)

fout->T      = T
fout->TOPOG  = ORO
```

Creating compressed NetCDF

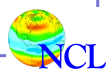
- Reduces file size
- Nine levels of compression possible
- Two steps required:

```
setfileoption("nc","Format","NetCDF4Classic")
setfileoption("nc","CompressionLevel",5)
```

- Recommend using compression level 1
- Does increase amount of time to read file
- Can get reductions of 90%
  - Fields like SST will have significant reduction
  - Random numbers, not so much

NCL File I/O Outline

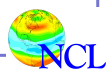
- Goals
- Data formats overview
- Tools overview
- Reading data
- Writing data
- Conclusion



Conclusion

LOOK AT YOUR DATA / KNOW YOUR DATA

- Trust the source
- Dump the file contents – ncl_filedump
- Look at metadata - printVarSummary
- Look at coordinates
- Print min/max - printMinMax
- Use other functions to examine data
- Plot the data with NCL or quick look tools



More conclusions

- Use multiple tools if necessary
- Be careful with large data
- Read the documentation ☺

Final questions?