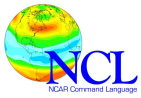


Basics II

NetCDF

IO

Contents

netCDF

- What is netCDF
- NetCDF conventions
- Examine a netCDF file
- NCL: read and plot a variable from a netCDF file

What is netCDF

netCDF (**N**etwork **C**ommon **D**ata **F**orm) from UNIDATA

is a self-describing, machine-independent data format

→ Contains all information within the file
→ No need of external information to determine file contents
→ Portable, readable on different machine architectures

Software tools:

- Analysis and Visualization: **NCL, GrADS, IDL, Matlab, Python, F, C, C++, Java,..**
- For calculation and manipulation issues: **CDO** and **NCO**
- Visualization (quick look): **ncview, panoply**

netCDF conventions

COARDS (stopped 1995)

- **C**ooperative **O**cean/**A**tmosphere **R**easearch **D**ata **S**ervice

CF current version: 1.6 (draft version 1.7)

- **C**limate and **F**orecast Metadata Convention
- Generalize and extend the COARDS convention
- except missing values are identified

Examine a netCDF file

Command line tools: **ncdump, ncview** and **ncl_filedump**

ncdump comes with the netCDF library package!

`ncdump file | less`	dump the file contents to stdout
`ncdump -h file | less`	dump the header information
`ncdump -v slp file | less`	dump the variable slp information
`ncdump -c file | less`	dump the coordinate data

ncview visualize the file contents

`ncview file`	open ncview GUI

ncl_filedump comes with NCL!

`ncl_filedump file`	similar to ncdump, but more general

NetCDF file (1)

ncdump -h ECHAM5_OM_A1B_2001_Jan_2D.nc

```
netcdf ECHAM5_OM_A1B_2001_Jan_2D {
dimensions:
        lon = 192 ;
        lat = 96 ;
        time = UNLIMITED ; // (124 currently)
variables:
        double lon(lon) ;
                lon:long_name = "longitude" ;
                lon:units = "degrees_east" ;
                lon:standard_name = "longitude" ;
                lon:axis = "X" ;
        double lat(lat) ;
                lat:long_name = "latitude" ;
                lat:units = "degrees_north" ;
                lat:standard_name = "latitude" ;
                lat:axis = "Y" ;
        double time(time) ;
                time:units = "hours since 2001-01-01 00:00:00" ;
                time:calendar = "standard" ;

… continuing
```

NetCDF file (2)

```
        float tsurf(time, lat, lon) ;
                tsurf:long_name = "surface temperature" ;
                tsurf:units = "K" ;
                tsurf:code = 169 ;
                tsurf:table = 128 ;
                tsurf:grid_type = "gaussian" ;
        float precip(time, lat, lon) ;
                precip:long_name = "total precipitation" ;
                precip:units = "kg/m^2s" ;
                precip:code = 4 ;
                precip:table = 128 ;
                precip:grid_type = "gaussian" ;
        float u10(time, lat, lon) ;
                u10:long_name = "10m u-velocity" ;
                u10:units = "m/s" ;
                u10:code = 165 ;
                u10:table = 128 ;
                u10:grid_type = "gaussian" ;
        float v10(time, lat, lon) ;
                v10:long_name = "10m v-velocity" ;
                v10:units = "m/s" ;
                v10:code = 166 ;
                v10:table = 128 ;
                v10:grid_type = "gaussian" ;
… continuing
```


NetCDF file (3)

```
// global attributes:
                :CDI = "Climate Data Interface version 1.5.0
(http://code.zmaw.de/projects/cdi)" ;
                :Conventions = "CF-1.0" ;
                :history = "Wed Mar 13 11:43:44 2013: cdo -r -f nc selmon,1
/work/kv0653//Tutorial_AvizoGreen/ECHAM5_OM_A1B_2001_2D.nc
ECHAM5_OM_A1B_2001_Jan_2D.nc\n",
                        "Wed Feb 27 10:05:05 2008: cdo merge
EH5_OM_A1B_1_TSURF_1-1460.nc EH5_OM_A1B_1_TPREC_1-1460.nc
EH5_OM_A1B_1_U10M_1-1460.nc EH5_OM_A1B_1_V10M_1-1460.nc EH5_OM_A1B_1_QVI_1-
1460.nc EH5_OM_A1B_1_MSLP_1-1460.nc
/scratch1/wrkshr/k203024/tutorial/ECHAM5_OM_A1B_2001_2D.nc\n",
                        "Fri Jun 08 11:23:40 2007: cdo -f nc -r
selindexbox,97,96,1,96 EH5_OM_A1B_1_MSLP_1-1460.grb EH5_OM_A1B_1_MSLP_1-
1460.nc" ;
                :source = "ECHAM5.2" ;
                :institution = "Max-Planck-Institute for Meteorology" ;
                :CDO = "Climate Data Operators version 1.5.0
(http://code.zmaw.de/projects/cdo)" ;
}
```


netCDF and NCL

→ netCDF, GRIB, HDF, HDF-EOS and Shapefile files are put into a consistent structure by NCL

Facilitation of

- Writing netCDF and HDF files
- Writing functions that add, change, query or use meta data

Some NCL functions use/access meta data interally

- **gsn_csm_*** graphics functions for labeling and map projection

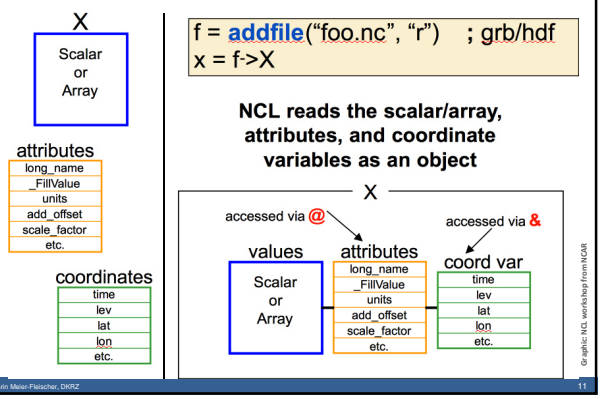
Data units and long_name as well as latitude and longitude units are taken from the netCDF file to annotate the plot and do some map presettings.

NCL: Read a netCDF file and variable

```
f      =  addfile("$NCL_TUT/data/rectilinear_grid_2D.nc","r")
tsurf  =  f->tsurf
printVarSummary(tsurf)

Variable: tsurf
Type: float
Total Size: 9142272 bytes
            2285568 values
Number of Dimensions: 3
Dimensions and sizes:  [time | 124] x [lat | 96] x [lon | 192]
Coordinates:
            time: [   0.. 738]
            lat: [88.57216851400727..-88.57216851400727]
            lon: [-180..178.125]
Number Of Attributes: 5
  long_name :  surface temperature
  units :      K
  code :       169
  table :      128
  grid_type :  gaussian
```


NCL: netCDF variable model



IO
Contents

- Supported formats
- Open single supported file
- Read ASCII files
- Read binary files
- Read CSV files
- Write netCDF
- Write ASCII
- Open multiple files

IO

Supported formats (1)

- **Import**
  - **netCDF3, netCDF4**
  - **HDF4-SDS, HDF4-EOS, HDF5, HDF5-EOS**
  - **GRIB1, GRIB2**
  - **CCM History Tape (outdated)**
  - **Shapefiles**
  - **Binary**
  - **ASCII**
- **Export**
  - **netCDF3, netCDF4**
  - **HDF4, HDF5**
  - **Binary**
  - **ASCII**

IO

Supported formats (2)

- **Graphics output**
  - **PS, EPS, EPSI**
  - **PDF**
  - **PNG**
  - **NCGM (outdated)**
  - **X11 (output to a X11 window)**

IO

NCL command line tools **ncl_filedump** and **ncl_convert2nc**

ncl_filedump

- reads any supported file's contents
- Input file name needs a file suffix as identifier, but the file doesn't have to e.g. file name on disc is **A210_day_mean**

```
ncl_filedump A210_day_mean.nc
```

→ add .nc to file name when calling ncl_filedump
→ identify file as netCDF file

ncl_convert2nc

- Converts GRIB, HDF, Shapefiles to netCDF
- Output file name is the basename plus .nc suffix

- `ncl_convert2nc mytest.grb` → create mytest.nc
- `ncl_convert2nc -L -nc4c -cl 1 mytest.grb`
  → -L (large file > 2GB)
  → -nc4c (netCDF4)
  → -cl 1 (compress. level 1)

NCL

IO

Read supported format

f = **addfile**(filename.ext, status)

filename	any valid file name; type string
ext	extension that identify the type of file; type string netCDF: "**nc** " or "**cdf** " HDF: "**hdf**", "**hdfeos**", "**h5**", "**he5**" GRIB: "**grb**", "**grib**" CCHMT: "**ccm**" Shape: "**shp**" extension not required to be attached to file
Status	"**r**" read, "**w**" write, "**c**" create
f	reference/pointer to a single file; any valid variable name; can have file attributes

NCL

IO

Open a single file:

```
f    = addfile("file01.nc", "r")

fout = addfile("/tmp/file01.nc", "w")
h    = addfile("$NCL_TUT/file01.hdf", "c")
s    = addfile("file01.shp", "r")
```

Numerous functions to query the contents of file (supported format)

- getfilevarnames
- getfilevardims
- getfileatts
- getfilevardimsizes
- getfilevartypes
- isfilevar
- isfilevaratt
- isfilevardim
- isfilevarcoord

NCL

IO

Import variable from supported file

```
f      = addfile("file01.nc", "r")
tsurf  = f->tsurf
```

-> Variable tsurf of file pointer f

→ read variable with all attributes (meta data)
→ NO space allowed left or right of ->
→ use $..$ to use NCL variable instead file variable

```
varName = "slp"      ;-- define an NCL variable
slp     = f->$varName$
```

Strip of meta data except _FillValue

```
tsurf  = (/f->tsurf/)
```

IO

Read ASCII data

asciiread - reads a file that contains ASCII representations of basic data types.

str_fields_count - count the number of fields in a string, given a delimiter.

str_get_cols - retrieve a particular column in a string, given a start and end index.

str_get_field - retrieve a particular field in a string, given a delimiter.

str_split_csv - splits strings into an array of strings based on a single delimiter.

str_sub_str - replace a substring with another substring.

readAsciiHead - reads an ASCII file and returns just the header.

numAsciiCol - returns the number of columns in an ASCII file.

numAsciiRow - returns the number of rows in an ASCII file.

IO

ASCII file (17 lines, 2 columns):

Estimated world population (in millions) taken from Wikipedia

1000 310
1750 791
1800 978
1850 1262
....

```
data = asciiread("asc2.txt",-1,"float")        ; 1D
```

To read this data into a **2D** array dimensioned **17 x 2** (17 rows by 2 columns), use:

```
data = asciiread("asc2.txt",(/17,2/),"float") ; 2D
year = data(:,0)
pnum = data(:,1)
```

(0)	1000
(1)	310
(2)	1750
(3)	791
(4)	1800
(5)	978
(6)	1850
(7)	1262
(8)	1900
(9)	1650
.....	

(0,0)	1000
(0,1)	310
(1,0)	1750
(1,1)	791
(2,0)	1800
(2,1)	978
(3,0)	1850
(3,1)	1262
(4,0)	1900
(4,1)	1650

IO

Read binary data

Direct Access: data = **fbindirread**(path,rec,dim,type)
Sequential Access: data = **fbinrecread**(path,rec,dim,type)
Cray (C block IO write): data = **cbinread**(path,dim,type)
Cray Sequential: data = **craybinrecread**(path,rec,dim,type)

NCL allows users to read binary files created e.g. on big-endian machines on little-endian machines and vice versa via the *setfileoption* procedure.

```
setfileoption("bin","ReadByteOrder","LittleEndian")
v = cbinread("data.littleEndian.bin",-1,"float")
```

IO

Read CSV (= comma-separated values) data

Example file:

```
34,67,56
36,87,78.1
31,56,88.4
29,67,92
54,71,68.9
42,65,82
```

```
lines = asciiread(filename,-1,"string")

delim  = ","
col1   = toint(str_get_field(lines,1,delim))
col2   = toint(str_get_field(lines,2,delim))
col3   = tofloat(str_get_field(lines,3,delim))
```


IO

Write netCDF file

```
f       = addfile("file01.nc", "r")
tsurf   = f->tsurf

ts      = tsurf                 ; copy variable with all
                                ; meta data
ts      = ts-273.15             ; convert to °C
ts@units = "degC"               ; set new units attribute

system("/bin/rm -f fout.nc")    ; delete if existing
ncdf  = addfile("fout.nc" ,"c")  ; create new output file

filedimdef(ncdf,"time",-1,True)  ; set time UNLIMITED dim

ncdf->tsurf  = ts               ; write ts to fout.nc
```


IO

Write ASCII file

print_table - formatted print of all elements from a list
write_table - writes formatted, mixed-type data with a single format statement.
write_matrix - writes nicely-formatted 2D arrays of integer, float, or double precision data.
asciiwrite - an older and rather limited function that writes one value per line. This is useful for outputting a one-dimensional time series.

```
data = random_uniform(-5,5,(/2,3,4/))   ; dummy 2x3x4 array
asciiwrite("file1.txt",data)            ; write to a file
```

→

```
-1.762895
-1.75608
-0.06612359
-2.112701
```

IO

Open multiple files

```
flist = systemfunc("ls file_*.nc")
f     = addfiles(flist, "r")

Choose how files are combined and read in variable across
files:

ListSetType (f, "cat")     ; concatenate or "merge" (default)
T = f[:]->T                ; note syntax [:]
```