COMP 696: Advanced Parallel Computing Note : Network Common Data Form (NetCDF) Overview

Mary Thomas

Department of Computer Science
Computational Science Research Center (CSRC)
San Diego State University (SDSU)

Posted: 10/19/15
Updated: 10/19/15

Table of Contents

Topics

- What is NetCDF
- References
- Compiling & Running NetCDF on tuckoo
- Creating NetCDF Files
- Reading NetCDF Files
- Viewing NetCDF File contents with *ncdump*
- Viewing NetCDF File contents with *ncview*

What is NetCDF

- netCDF is an interface to a library of data access functions for storing and retrieving data in the form of arrays
- netCDF is an abstraction that supports a view of data as a collection of self-describing, portable objects that can be accessed through a simple interface.
- Array values may be accessed directly, without knowing details of how the data are stored.
- Auxiliary information about the data, such as what units are used, may be stored with the data.
- Data encoding: XDR (eXternal Data Representation) supports network-transparency (machine-independence).
- Need to be aware of what format you are using.

NetCDF is not a Database System

- relational database software is not suitable for the kinds of data access supported by the netCDF interface
- relational models do not support multidimensional objects (arrays) as a basic unit of data access (esp. arrays)
- have generally poor performance on large arrays: e.g. collections of satellite images, scientific model outputs and long-term global weather observations are beyond the capabilities of most database systems to organize and index for efficient retrieval.
- many database utilities are not needed and add overhead

NetCDF File Formats

- NetCDF Classic Format (V3.6)
  - File format is binary: original version, also default
  - Identified using four bytes, CDF\001in file header.
  - format is identical to the format used by every previous version of netCDF –¿ maximum portability,
- NetCDF 64-bit Offset Format
  - ”64-bit offset” supports creation of larger files.
  - Files identified with a CDF \ 002 at the beginning of the file.
  - Introduced in version 3.6.0, earlier versions can’t read 64-bit offset files.
- NetCDF-4 Format: HDF5
  - Features include groups, compound types, variable length arrays, new unsigned integer types, parallel I/O access, etc. None of these new features can be used with classic or 64-bit offset files.
  - NetCDF-4 files can’t be created at all, unless the netCDF configure script is run with enable-netcdf-4. Requires version 1.8.0 of HDF5.
  - Only have C and Fortran interfaces.
  - Interoperable with HDF5, version 1.8.0 or better).

Components of an NetCDF Dataset

- Data Model: How NetCDF Sees Data – dimensions, variables, and attributes, which all have both a name and an ID number
- Dimensions: Specifying Data Shape – used to represent a real physical dimension
- Variables: Storing Data – Variables are used to store the bulk of the data in a netCDF dataset. A variable represents an array of values of the same type.
- Attributes: storing metadata about the data

NetCDF References

- https://www.unidata.ucar.edu/software/netcdf/docs/netcdf-tutorial.html#Intro
- https://www.unidata.ucar.edu/software/netcdf/docs/netcdf.html#Top
- NetCDF C Ref Guide https://www.unidata.ucar.edu/software/netcdf/docs/netcdf-c.html
- Viz tools:
  - ncview
  - http://www.epic.noaa.gov/java/ncBrowse/
  - NASA Panoply http://www.giss.nasa.gov/tools/panoply/

NetCDF makefile

```
1  MPIF90 = mpif90
2  MPICC = mpicc
3  CC    = gcc
4
5  ### tuckoo
6  #NETCDF_LIB = -L/usr/lib -lnetcdf -lnetcdff
7  #NETCDF_INC = -I/usr/include
8  ### gidget
9  NETCDF_LIB = -L/opt/local/lib -lnetcdf -lnetcdff
10 NETCDF_INC = -I/opt/local/include
11
12 all: simple_xy_wr simple_xy_rd sfc_pres_temp_wr sfc_pres_temp_rd pres_temp_4D_wr pres_temp_4D_rd
13
14 simple_xy_rd: simple_xy_rd.c
15 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o simple_xy_rd simple_xy_rd.c
16
17 simple_xy_wr: simple_xy_wr.c
18 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o simple_xy_wr simple_xy_wr.c
19
20 sfc_pres_temp_wr: sfc_pres_temp_wr.c
21 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o sfc_pres_temp_wr sfc_pres_temp_wr.c
22
23 sfc_pres_temp_rd: sfc_pres_temp_rd.c
24 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o sfc_pres_temp_rd sfc_pres_temp_rd.c
25
26 pres_temp_4D_wr: pres_temp_4D_wr.c
27 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o pres_temp_4D_wr pres_temp_4D_wr.c
28
29 pres_temp_4D_rd: pres_temp_4D_rd.c
30 $(MPICC) $(NETCDF_LIB) -lnetcdf -lnetcdff $(NETCDF_INC) -o pres_temp_4D_rd pres_temp_4D_rd.c
31
32 clean:
33 rm -rf *.o simple_xy_wr simple_xy_rd sfc_pres_temp_wr sfc_pres_temp_rd \
34     pres_temp_4D_wr pres_temp_4D_rd
```

Template for Creating or Modifying NetCDF Files

```
1
2         nc_open              /* open existing netCDF dataset */
3            ...
4          nc_redef             /* put it into define mode */
5              ...
6            nc_def_dim         /* define additional dimensions (if any) */
7              ...
8            nc_def_var         /* define additional variables (if any) */
9              ...
10           nc_put_att         /* define additional attributes (if any) */
11             ...
12         nc_enddef            /* check definitions, leave define mode */
13             ...
14           nc_put_var         /* provide values for new variables */
15             ...
16         nc_close             /* close netCDF dataset */
```

Creating/Writing NetCDF File

- int nc_create (const char* path, int cmode, int *ncidp);
- int nc_def_dim(int ncid, const char name, size_t len, int dimidp);
- int nc_def_var (int ncid, const char *name, nc_type xtype, int ndims, const int dimids[], int *varidp);
- int nc_put_att_text(int ncid, int varid, const char *name, size_t len, const char *tp)
- int nc_enddef(int ncid)
- int nc_put_var_int(int ncid, int varid, const int *ip)
- int nc_put_var_float(int ncid, int varid, const float *fp)
- int nc_put_vara_float(int ncid, int varid, const size_t start[], const size_t count[], const float *fp)
- int nc_close(int ncid)

Template for Reading NetCDF Files

```
1         nc_open                  /* open existing netCDF dataset */
2           ...
3         nc_inq                   /* find out what is in it */
4              ...
5            nc_inq_dim            /* get dimension names, lengths */
6              ...
7            nc_inq_var            /* get variable names, types, shapes */
8                 ...
9               nc_inq_attname     /* get attribute names */
10                ...
11              nc_inq_att         /* get attribute types and lengths */
12                ...
13              nc_get_att         /* get attribute values */
14                ...
15           nc_get_var            /* get values of variables */
16             ...
17        nc_close                 /* close netCDF dataset */
```

Reading NetCDF File

- int nc_open(int ncid)
- int nc_inq_varid(int ncid, int *nvars, int *varids)
- int nc_get_varid(int ncid, const char *name, int *varidp)
- int nc_get_var_int(int ncid, int varid, int *ip)
- int nc_get_var_float(int ncid, int varid, float *fp)
- int nc_get_att_text(int ncid, int varid, const char *name, char *tp)
- int nc_close(int ncid)

Write NetCDF Data File: simple_xy_wr.c

```
1  /* This is part of the netCDF package.
2        Copyright 2006 University Corporation for Atmospheric Research/Unidata.
3        See COPYRIGHT file for conditions of use.
4
5        This is a very simple example which writes a 2D array of
6        sample data. To handle this in netCDF we create two shared
7        dimensions, "x" and "y", and a netCDF variable, called "data".
8
9        This example demonstrates the netCDF C API. This is part of the
10       netCDF tutorial, which can be found at:
11       http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-tutorial
12
13       Full documentation of the netCDF C API can be found at:
14       http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-c
15
16       $Id: simple_xy_wr.c,v 1.12 2007/02/14 20:59:21 ed Exp $
17    */
18    #include <stdlib.h>
19    #include <stdio.h>
20    #include <netcdf.h>
21
22    /* This is the name of the data file we will create. */
23    #define FILE_NAME "simple_xy.nc"
24
25    /* We are writing 2D data, a 6 x 12 grid. */
26    #define NDIMS 2
27    #define NX 6
28    #define NY 12
29
30    /* Handle errors by printing an error message and exiting with a
31     * non-zero status. */
32    #define ERRCODE 2
33    #define ERR(e) {printf("Error: %s\n", nc_strerror(e)); exit(ERRCODE);}
```

simple_xy_wr.c

```
1   int     main()     {
2      /* When we create netCDF variables and dimensions, we get back an
3       * ID for each one. */
4      int ncid, x_dimid, y_dimid, varid;
5      int dimids[NDIMS];
6
7      /* This is the data array we will write. It will be filled with a
8       * progression of numbers for this example. */
9      int data_out[NX][NY];
10
11     /* Loop indexes, and error handling. */
12     int x, y, retval;
13
14     /* Create some pretend data. If this wasn't an example program, we
15      * would have some real data to write, for example, model
16      * output. */
17     for (x = 0; x < NX; x++)
18        for (y = 0; y < NY; y++)
19       data_out[x][y] = x * NY + y;
20
21     /* Always check the return code of every netCDF function call. In
22      * this example program, any retval which is not equal to NC_NOERR
23      * (0) will cause the program to print an error message and exit
24      * with a non-zero return code. */
25
26     /* Create the file. The NC_CLOBBER parameter tells netCDF to
27      * overwrite this file, if it already exists.*/
28     if ((retval = nc_create(FILE_NAME, NC_CLOBBER, &ncid)))
29        ERR(retval);
30
31     /* Define the dimensions. NetCDF will hand back an ID for each. */
32     if ((retval = nc_def_dim(ncid, "x", NX, &x_dimid)))
33        ERR(retval);
34     if ((retval = nc_def_dim(ncid, "y", NY, &y_dimid)))
35        ERR(retval);
```

simple_xy_wr.c

```
1         /* The dimids array is used to pass the IDs of the dimensions of
2          * the variable. */
3         dimids[0] = x_dimid;
4         dimids[1] = y_dimid;
5
6         /* Define the variable. The type of the variable in this case is
7          * NC_INT (4-byte integer). */
8         if ((retval = nc_def_var(ncid, "data", NC_INT, NDIMS,
9           dimids, &varid)))
10           ERR(retval);
11
12        /* End define mode. This tells netCDF we are done defining
13         * metadata. */
14        if ((retval = nc_enddef(ncid)))
15           ERR(retval);
16
17        /* Write the pretend data to the file. Although netCDF supports
18         * reading and writing subsets of data, in this case we write all
19         * the data in one operation. */
20        if ((retval = nc_put_var_int(ncid, varid, &data_out[0][0])))
21           ERR(retval);
22
23        /* Close the file. This frees up any internal netCDF resources
24         * associated with the file, and flushes any buffers. */
25        if ((retval = nc_close(ncid)))
26           ERR(retval);
27
28        printf("*** SUCCESS writing example file simple_xy.nc!\n");
29        return 0;
```

Read NetCDF Data File: simple_xy_rd.c

```
1   #include <stdlib.h>
2   #include <stdio.h>
3   #include <netcdf.h>
4
5   /* This is the name of the data file we will read. */
6   #define FILE_NAME "simple_xy.nc"
7
8   /* We are reading 2D data, a 6 x 12 grid. */
9   #define NX 6
10  #define NY 12
11
12  /* Handle errors by printing an error message and exiting with a
13   * non-zero status. */
14  #define ERRCODE 2
15  #define ERR(e) {printf("Error: %s\n", nc_strerror(e)); exit(ERRCODE);}
16
17  int     main()  {
18     /* This will be the netCDF ID for the file and data variable. */
19     int ncid, varid;
20     int data_in[NX][NY];
21
22     /* Loop indexes, and error handling. */
23     int x, y, retval;
24
25     /* Open the file. NC_NOWRITE tells netCDF we want read-only access
26      * to the file.*/
27     if ((retval = nc_open(FILE_NAME, NC_NOWRITE, &ncid)))
28         ERR(retval);
29
30     /* Get the varid of the data variable, based on its name. */
31     if ((retval = nc_inq_varid(ncid, "data", &varid)))
32        ERR(retval);
```

simple_xy_rd.c

```
1
2         /* Read the data. */
3         if ((retval = nc_get_var_int(ncid, varid, &data_in[0][0])))
4            ERR(retval);
5
6         /* Check the data. */
7         for (x = 0; x < NX; x++)
8            for (y = 0; y < NY; y++)
9               if (data_in[x][y] != x * NY + y)
10               return ERRCODE;
11
12        /* Check the data. */
13        printf("Data:\n");
14        for (x = 0; x < NX; x++) {
15           for (y = 0; y < NY; y++)
16               printf("%d ",data_in[x][y]);
17           printf("\n");
18        }
19        /* Close the file, freeing all resources. */
20        if ((retval = nc_close(ncid)))
21           ERR(retval);
22
23        printf("*** SUCCESS reading example file %s!\n", FILE_NAME);
24        return 0;
25    }
```

NetCDF: common ncdump commands

- ncdump h ficname.nc
  Shows only the header of the NetCDF file
- ncdump v varname ficname.nc
  Shows one variable of the file
- ncdump t var TIME ficname.nc
  Shows station time as date time string
- ncdump t var TIME ficname.nc
  Shows station time as ISO-8601 string

File Contents (ncdump)

```
1
2  [gidget:netcdf/ser/simple] mthomas% ./simple_xy_wr
3  *** SUCCESS writing example file simple_xy.nc!
4
5  [gidget:netcdf/ser/simple] mthomas% ncdump simple_xy.nc
6
7  netcdf simple_xy {
8  dimensions:
9  x = 6 ;
10 y = 12 ;
11 variables:
12 int data(x, y) ;
13 data:
14
15  data =
16   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
17   12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
18   24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
19   36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
20   48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
21   60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71 ;
22 }
```

More Complex NetCDF File: sfc_pres_temp_wr.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <netcdf.h>
4
5  /* This is the name of the data file we will create. */
6  #define FILE_NAME "sfc_pres_temp.nc"
7
8  /* We are writing 2D data, a 6 x 12 lat-lon grid. We will need two
9   * netCDF dimensions. */
10 #define NDIMS 2
11 #define NLAT 6
12 #define NLON 12
13 #define LAT_NAME "latitude"
14 #define LON_NAME "longitude"
15
16 /* Names of things. */
17 #define PRES_NAME "pressure"
18 #define TEMP_NAME "temperature"
19 #define UNITS "units"
20 #define DEGREES_EAST "degrees_east"
21 #define DEGREES_NORTH "degrees_north"
22
23 /* These are used to construct some example data. */
24 #define SAMPLE_PRESSURE 900
25 #define SAMPLE_TEMP 9.0
26 #define START_LAT 25.0
27 #define START_LON -125.0
28
29 /* Handle errors by printing an error message and exiting with a
30  * non-zero status. */
31 #define ERR(e) {printf("Error: %s\n", nc_strerror(e)); return 2;}
```

sfc_pres_temp_wr.c

```
1   int    main()   {
2     int ncid, lon_dimid, lat_dimid, pres_varid, temp_varid;
3
4   /* In addition to the latitude and longitude dimensions, we will also
5      create latitude and longitude netCDF variables which will hold the
6      actual latitudes and longitudes. Since they hold data about the
7      coordinate system, the netCDF term for these is: "coordinate variables." */
8     int lat_varid, lon_varid;
9     int dimids[NDIMS];
10
11    /* We will write surface temperature and pressure fields. */
12    float pres_out[NLAT][NLON];
13    float temp_out[NLAT][NLON];
14    float lats[NLAT], lons[NLON];
15
16    /* It's good practice for each netCDF variable to carry a "units"
17     * attribute. */
18    char pres_units[] = "hPa";
19    char temp_units[] = "celsius";
20
21    int lat, lon;    /* Loop indexes. */
22    int retval;     /* Error handling. */
23
24    /* Create some pretend data. If this wasn't an example program, we
25     * would have some real data to write, for example, model output. */
26    for (lat = 0; lat < NLAT; lat++)
27       lats[lat] = START_LAT + 5.*lat;
28    for (lon = 0; lon < NLON; lon++)
29       lons[lon] = START_LON + 5.*lon;
30
31    for (lat = 0; lat < NLAT; lat++)
32       for (lon = 0; lon < NLON; lon++)  {
33        pres_out[lat][lon] = SAMPLE_PRESSURE + (lon * NLAT + lat);
34        temp_out[lat][lon] = SAMPLE_TEMP + .25 * (lon * NLAT + lat);
35       }
```

sfc_pres_temp_wr.c

```
1         /* Create the file. */
2         if ((retval = nc_create(FILE_NAME, NC_CLOBBER, &ncid)))
3            ERR(retval);
4
5         /* Define the dimensions. */
6         if ((retval = nc_def_dim(ncid, LAT_NAME, NLAT, &lat_dimid)))
7            ERR(retval);
8         if ((retval = nc_def_dim(ncid, LON_NAME, NLON, &lon_dimid)))
9            ERR(retval);
10
11        /* Define coordinate netCDF variables. They will hold the
12           coordinate information, that is, the latitudes and longitudes. A
13           varid is returned for each.*/
14        if ((retval = nc_def_var(ncid, LAT_NAME, NC_FLOAT, 1, &lat_dimid,
15          &lat_varid)))
16           ERR(retval);
17        if ((retval = nc_def_var(ncid, LON_NAME, NC_FLOAT, 1, &lon_dimid,
18          &lon_varid)))
19           ERR(retval);
20
21        /* Define units attributes for coordinate vars. This attaches a
22           text attribute to each of the coordinate variables, containing
23           the units. Note that we are not writing a trailing NULL, just
24           "units", because the reading program may be fortran which does
25           not use null-terminated strings. In general it is up to the
26           reading C program to ensure that it puts null-terminators on
27           strings where necessary.*/
28        if ((retval = nc_put_att_text(ncid, lat_varid, UNITS,
29       strlen(DEGREES_NORTH), DEGREES_NORTH)))
30           ERR(retval);
31        if ((retval = nc_put_att_text(ncid, lon_varid, UNITS,
32       strlen(DEGREES_EAST), DEGREES_EAST)))
33           ERR(retval);
```

sfc_pres_temp_wr.c

```
1         /* Define the netCDF variables. The dimids array is used to pass
2            the dimids of the dimensions of the variables.*/
3         dimids[0] = lat_dimid;
4         dimids[1] = lon_dimid;
5         if ((retval = nc_def_var(ncid, PRES_NAME, NC_FLOAT, NDIMS, dimids, &pres_varid)))
6            ERR(retval);
7         if ((retval = nc_def_var(ncid, TEMP_NAME, NC_FLOAT, NDIMS, dimids, &temp_varid)))
8            ERR(retval);
9
10        /* Define units attributes for vars. */
11        if ((retval = nc_put_att_text(ncid, pres_varid, UNITS, strlen(pres_units), pres_units)))
12           ERR(retval);
13        if ((retval = nc_put_att_text(ncid, temp_varid, UNITS, strlen(temp_units), temp_units)))
14           ERR(retval);
15
16        /* End define mode. */
17        if ((retval = nc_enddef(ncid)))    ERR(retval);
18
19        /* Write the coordinate variable data. This will put the latitudes
20           and longitudes of our data grid into the netCDF file. */
21        if ((retval = nc_put_var_float(ncid, lat_varid, &lats[0])))      ERR(retval);
22        if ((retval = nc_put_var_float(ncid, lon_varid, &lons[0])))    ERR(retval);
23
24        /* Write the pretend data. This will write our surface pressure and
25           surface temperature data. The arrays of data are the same size
26           as the netCDF variables we have defined. */
27        if ((retval = nc_put_var_float(ncid, pres_varid, &pres_out[0][0])))      ERR(retval);
28        if ((retval = nc_put_var_float(ncid, temp_varid, &temp_out[0][0])))   ERR(retval);
29
30        /* Close the file. */
31        if ((retval = nc_close(ncid)))  ERR(retval);
32
33        printf("***␣SUCCESS␣writing␣example␣file␣sfc_pres_temp.nc!\n");
34        return 0;
35    }
```

More Complex NetCDF File: sfc_pres_temp_rd.c

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <netcdf.h>
4
5   /* This is the name of the data file we will read. */
6   #define FILE_NAME "sfc_pres_temp.nc"
7
8   /* We are reading 2D data, a 6 x 12 lat-lon grid. */
9   #define NDIMS 2
10  #define NLAT 6
11  #define NLON 12
12
13  #define LAT_NAME "latitude"
14  #define LON_NAME "longitude"
15  #define PRES_NAME "pressure"
16  #define TEMP_NAME "temperature"
17
18  /* These are used to calculate the values we expect to find. */
19  #define SAMPLE_PRESSURE 900
20  #define SAMPLE_TEMP 9.0
21  #define START_LAT 25.0
22  #define START_LON -125.0
23
24  /* For the units attributes. */
25  #define UNITS "units"
26  #define PRES_UNITS "hPa"
27  #define TEMP_UNITS "celsius"
28  #define LAT_UNITS "degrees_north"
29  #define LON_UNITS "degrees_east"
30  #define MAX_ATT_LEN 80
31
32  /* Handle errors by printing an error message and exiting with a
33   * non-zero status. */
34  #define ERR(e) {printf("Error: %s\n", nc_strerror(e)); return 2;}
```

sfc_pres_temp_rd.c

```
1   int     main()   {
2      int ncid, pres_varid, temp_varid;
3      int lat_varid, lon_varid;
4
5      /* We will read surface temperature and pressure fields. */
6      float pres_in[NLAT][NLON];
7      float temp_in[NLAT][NLON];
8
9      /* For the lat lon coordinate variables. */
10     float lats_in[NLAT], lons_in[NLON];
11
12     /* To check the units attributes. */
13     char pres_units_in[MAX_ATT_LEN], temp_units_in[MAX_ATT_LEN];
14     char lat_units_in[MAX_ATT_LEN], lon_units_in[MAX_ATT_LEN];
15
16     /* We will learn about the data file and store results in these
17        program variables. */
18     int ndims_in, nvars_in, ngatts_in, unlimdimid_in;
19
20     int lat, lon;             /* Loop indexes. */
21
22     int retval;        /* Error handling. */
23
24     /* Open the file. */
25     if ((retval = nc_open(FILE_NAME, NC_NOWRITE, &ncid)))     ERR(retval);
26
27     /* There are a number of inquiry functions in netCDF which can be
28        used to learn about an unknown netCDF file. NC_INQ tells how
29        many netCDF variables, dimensions, and global attributes are in
30        the file; also the dimension id of the unlimited dimension, if
31        there is one. */
32     if ((retval = nc_inq(ncid, &ndims_in, &nvars_in, &ngatts_in, &unlimdimid_in)))
33          ERR(retval);
34
35     /* In this case we know that there are 2 netCDF dimensions, 4
36        netCDF variables, no global attributes, and no unlimited  dimension. */
37     if (ndims_in != 2 || nvars_in != 4 || ngatts_in != 0 ||
38         unlimdimid_in != -1) return 2;
```

sfc_pres_temp_rd.c

```
1         /* Get the varids of the latitude and longitude coordinate
2          * variables. */
3         if ((retval = nc_inq_varid(ncid, LAT_NAME, &lat_varid)))     ERR(retval);
4         if ((retval = nc_inq_varid(ncid, LON_NAME, &lon_varid))     ERR(retval);
5
6         /* Read the coordinate variable data. */
7         if ((retval = nc_get_var_float(ncid, lat_varid, &lats_in[0])))   ERR(retval);
8         if ((retval = nc_get_var_float(ncid, lon_varid, &lons_in[0])))  ERR(retval);
9
10        /* Check the coordinate variable data. */
11        for (lat = 0; lat < NLAT; lat++)
12           if (lats_in[lat] != START_LAT + 5.*lat)
13               return 2;
14        for (lon = 0; lon < NLON; lon++)
15           if (lons_in[lon] != START_LON + 5.*lon)
16              return 2;
17
18        /* Get the varids of the pressure and temperature netCDF variables. */
19        if ((retval = nc_inq_varid(ncid, PRES_NAME, &pres_varid)))     ERR(retval);
20        if ((retval = nc_inq_varid(ncid, TEMP_NAME, &temp_varid)))    ERR(retval);
21
22        /* Read the data. Since we know the contents of the file we know
23         * that the data arrays in this program are the correct size to
24         * hold all the data. */
25        if ((retval = nc_get_var_float(ncid, pres_varid, &pres_in[0][0])))     ERR(retval);
26        if ((retval = nc_get_var_float(ncid, temp_varid, &temp_in[0][0])))    ERR(retval);
27
28        /* Check the data. */
29        for (lat = 0; lat < NLAT; lat++)
30           for (lon = 0; lon < NLON; lon++)
31       if (pres_in[lat][lon] != SAMPLE_PRESSURE + (lon * NLAT + lat) ||
32           temp_in[lat][lon] != SAMPLE_TEMP + .25 * (lon * NLAT + lat))
33                return 2;
```

sfc_pres_temp_rd.c

```
1
2         /* Each of the netCDF variables has a "units" attribute. Let's read
3            them and check them. */
4         if ((retval = nc_get_att_text(ncid, lat_varid, UNITS, lat_units_in)))  ERR(retval);
5         if (strncmp(lat_units_in, LAT_UNITS, strlen(LAT_UNITS)))
6                 return 2;
7
8         if ((retval = nc_get_att_text(ncid, lon_varid, UNITS, lon_units_in)))   ERR(retval);
9         if (strncmp(lon_units_in, LON_UNITS, strlen(LON_UNITS)))    return 2;
10
11        if ((retval = nc_get_att_text(ncid, pres_varid, UNITS, pres_units_in)))    ERR(retval);
12        if (strncmp(pres_units_in, PRES_UNITS, strlen(PRES_UNITS)))    return 2;
13
14        if ((retval = nc_get_att_text(ncid, temp_varid, UNITS, temp_units_in)))    ERR(retval);
15        if (strncmp(temp_units_in, TEMP_UNITS, strlen(TEMP_UNITS))) return 2;
16
17        /* Close the file. */
18        if ((retval = nc_close(ncid)))   ERR(retval);
19
20        printf("***␣SUCCESS␣reading␣example␣file␣sfc_pres_temp.nc!\n");
21        return 0;
22    }
```

Viewing NetCDF Data File Contents: ncdump command

```
[gidget:netcdf/ser/simple] mthomas% ncdump sfc_pres_temp.nc
netcdf sfc_pres_temp {
dimensions:
latitude = 6 ;
longitude = 12 ;
variables:
float latitude(latitude) ;
latitude:units = "degrees_north" ;
float longitude(longitude) ;
longitude:units = "degrees_east" ;
float pressure(latitude, longitude) ;
pressure:units = "hPa" ;
float temperature(latitude, longitude) ;
temperature:units = "celsius" ;
data:

 latitude = 25, 30, 35, 40, 45, 50 ;

 longitude = -125, -120, -115, -110, -105, -100, -95, -90, -85, -80, -75, -70 ;

 pressure =
  900, 906, 912, 918, 924, 930, 936, 942, 948, 954, 960, 966,
  901, 907, 913, 919, 925, 931, 937, 943, 949, 955, 961, 967,
  902, 908, 914, 920, 926, 932, 938, 944, 950, 956, 962, 968,
  903, 909, 915, 921, 927, 933, 939, 945, 951, 957, 963, 969,
  904, 910, 916, 922, 928, 934, 940, 946, 952, 958, 964, 970,
  905, 911, 917, 923, 929, 935, 941, 947, 953, 959, 965, 971 ;

 temperature =
  9, 10.5, 12, 13.5, 15, 16.5, 18, 19.5, 21, 22.5, 24, 25.5,
  9.25, 10.75, 12.25, 13.75, 15.25, 16.75, 18.25, 19.75, 21.25, 22.75, 24.25,
    25.75,
  9.5, 11, 12.5, 14, 15.5, 17, 18.5, 20, 21.5, 23, 24.5, 26,
  9.75, 11.25, 12.75, 14.25, 15.75, 17.25, 18.75, 20.25, 21.75, 23.25, 24.75,
    26.25,
  10, 11.5, 13, 14.5, 16, 17.5, 19, 20.5, 22, 23.5, 25, 26.5,
  10.25, 11.75, 13.25, 14.75, 16.25, 17.75, 19.25, 20.75, 22.25, 23.75,
    25.25, 26.75 ;
}
```

NetCDF File with time data: pres_temp_4D_rd.c

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <netcdf.h>
4
5   /* This is the name of the data file we will create. */
6   #define FILE_NAME "pres_temp_4D.nc"
7
8   /* We are writing 4D data, a 2 x 6 x 12 lvl-lat-lon grid, with 2
9      timesteps of data. */
10  #define NDIMS 4
11  #define NLAT 6
12  #define NLON 12
13  #define LAT_NAME "latitude"
14  #define LON_NAME "longitude"
15  #define NREC 2
16  #define REC_NAME "time"
17  #define LVL_NAME "level"
18  #define NLVL 2
19
20  /* Names of things. */
21  #define PRES_NAME "pressure"
22  #define TEMP_NAME "temperature"
23  #define UNITS "units"
24  #define DEGREES_EAST "degrees_east"
25  #define DEGREES_NORTH "degrees_north"
26
27  /* These are used to construct some example data. */
28  #define SAMPLE_PRESSURE 900
29  #define SAMPLE_TEMP 9.0
30  #define START_LAT 25.0
31  #define START_LON -125.0
32
33  /* For the units attributes. */
34  #define UNITS "units"
35  #define PRES_UNITS "hPa"
36  #define TEMP_UNITS "celsius"
37  #define LAT_UNITS "degrees_north"
38  #define LON_UNITS "degrees_east"
39  #define MAX_ATT_LEN 80
```

NetCDF File with time data: pres_temp_4D_rd.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <netcdf.h>
4
5  /* This is the name of the data file we will read. */
6  #define FILE_NAME "pres_temp_4D.nc"
7
8  /* We are reading 4D data, a 2 x 6 x 12 lvl-lat-lon grid, with 2
9     timesteps of data. */
10 #define NDIMS 4
11 #define NLAT 6
12 #define NLON 12
13 #define LAT_NAME "latitude"
14 #define LON_NAME "longitude"
15 #define NREC 2
16 #define REC_NAME "time"
17 #define LVL_NAME "level"
18 #define NLVL 2
19
20 /* Names of things. */
21 #define PRES_NAME "pressure"
22 #define TEMP_NAME "temperature"
23 #define UNITS "units"
24 #define DEGREES_EAST "degrees_east"
25 #define DEGREES_NORTH "degrees_north"
26
27 /* These are used to calculate the values we expect to find. */
28 #define SAMPLE_PRESSURE 900
29 #define SAMPLE_TEMP 9.0
30 #define START_LAT 25.0
31 #define START_LON -125.0
32
33 /* For the units attributes. */
34 #define UNITS "units"
35 #define PRES_UNITS "hPa"
36 #define TEMP_UNITS "celsius"
37 #define LAT_UNITS "degrees_north"
38 #define LON_UNITS "degrees_east"
39 #define MAX_ATT_LEN 80
```

Viewing NetCDF Data File Contents: ncdump command

```
[gidget:netcdf/ser/simple] mthomas% ./pres_temp_4D_wr
*** SUCCESS writing example file pres_temp_4D.nc!
[gidget:netcdf/ser/simple] mthomas% ncdump pres_temp_4D.nc
netcdf pres_temp_4D {
dimensions:
level = 2 ;
latitude = 6 ;
longitude = 12 ;
time = UNLIMITED ; // (2 currently)
variables:
float latitude(latitude) ;
latitude:    units = "degrees_north" ;
float longitude(longitude) ;
longitude:   units = "degrees_east" ;
float pressure(time, level, latitude, longitude) ;
pressure:   units = "hPa" ;
float temperature(time, level, latitude, longitude) ;
temperature:    units = "celsius" ;
data:

 latitude = 25, 30, 35, 40, 45, 50 ;

 longitude = -125, -120, -115, -110, -105, -100, -95, -90, -85, -80, -75, -70 ;

}
```

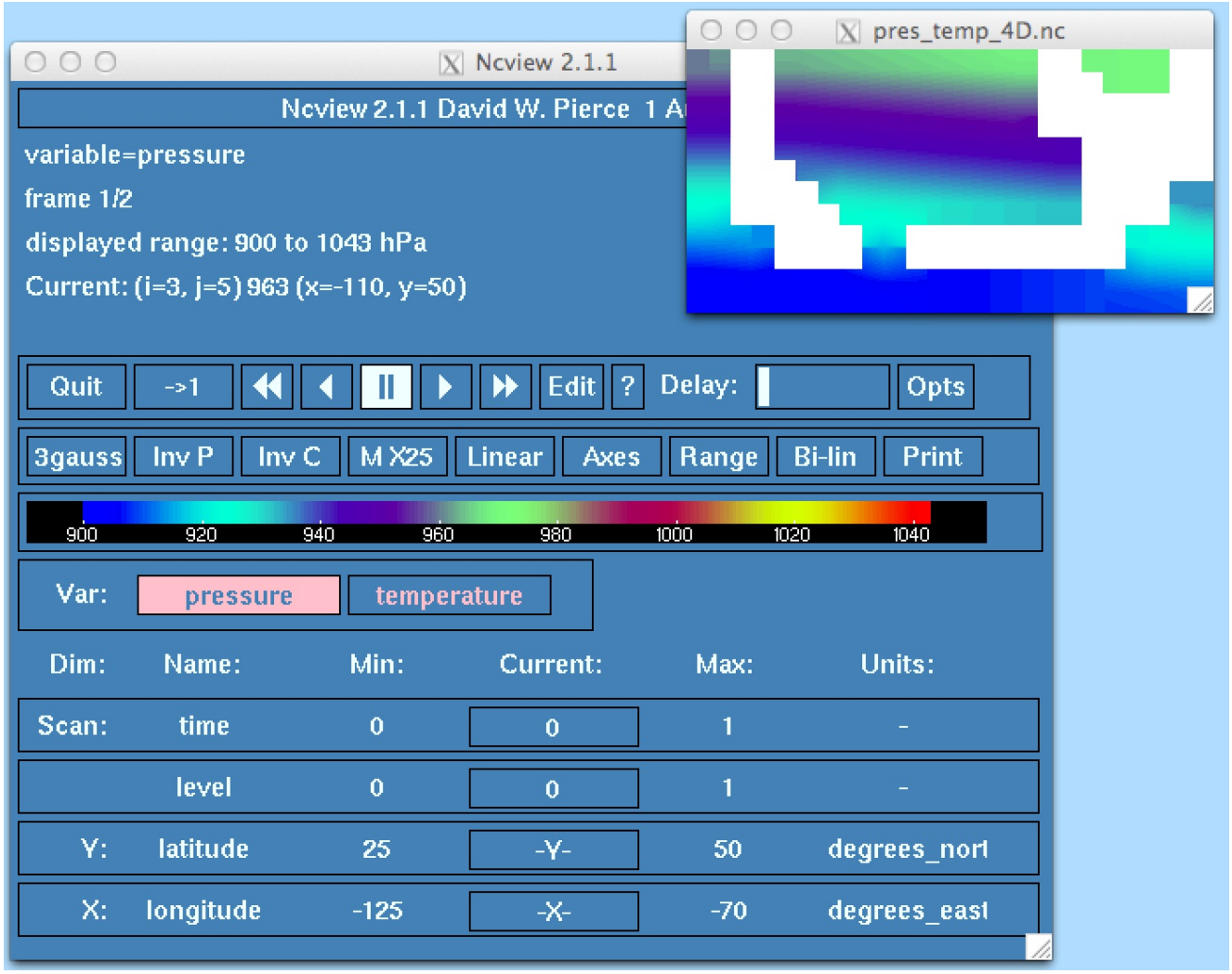
ncdump pres_temp_4D.nc

```
pressure =
 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911,
 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923,
 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,
 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947,
 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959,
 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971,
 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983,
 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007,
 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019,
 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031,
 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043,
 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911,
 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923,
 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,
 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947,
 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959,
 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971,
 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983,
 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995,
 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007,
 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019,
 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031,
 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043 ;
```

ncdump pres_temp_4D.nc

```
  temperature =
  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
  21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
  33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
  45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
  57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
  69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
  81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
  93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104,
  105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
  117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
  129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
  141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,
  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
  21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
  33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
  45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56,
  57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
  69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
  81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
  93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104,
  105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
  117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
  129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
  141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152 ;
}
```

Viewing NetCDF File Contents Using ncview



Viewing NetCDF Data File Contents: ncdump command

```
[gidget:dev/serucoam] mthomas% ls gcom_restart.nc
-rw-r--r--  1 mthomas  staff  406394044 Jul 30 15:04 gcom_restart.nc
[gidget:dev/serucoam] mthomas% !ncdump
ncdump -c gcom_restart.nc
netcdf gcom_restart {
dimensions:
time = UNLIMITED ; // (49 currently)
nxp1 = 97 ;
nyp1 = 33 ;
nzp1 = 33 ;
nx = 96 ;
ny = 32 ;
nz = 32 ;
variables:
double time(time) ;
time:units = "days since 1980-01-01 00:00:00" ;
time:long_name = "time" ;
time:calendar = "gregorian" ;
double u(time, nz, ny, nxp1) ;
u:units = "m/s" ;
u:long_name = "u component of the velocity" ;
u:coordinates = "ulon ulat ulev time" ;
double ulon(nz, ny, nxp1) ;
ulon:axis = "X" ;
ulon:units = "degrees_east" ; ulon:long_name = "longitude" ;
ulon:comment = "longitudes of u component velocity" ;
double ulat(nz, ny, nxp1) ;
ulat:axis = "Y" ;
ulat:units = "degrees_north" ; ulat:long_name = "latitude" ;
ulat:comment = "latitudes of u component velocity" ;
double ulev(nz, ny, nxp1) ;
ulev:axis = "Z" ;
ulev:positive = "up" ; ulev:units = "meters" ;
ulev:long_name = "depth" ;
ulev:comment = "depths of u component velocity" ;
```

ncdump gcom_restart.nc

```
1  double v(time, nz, nyp1, nx) ;
2  v:units = "m/s" ;
3  v:long_name = "v component of the velocity" ;
4  v:coordinates = "vlon vlat vlev time" ;
5  double vlon(nz, nyp1, nx) ;
6  vlon:axis = "X" ; vlon:units = "degrees_east" ;
7  vlon:long_name = "longitude" ;
8  vlon:comment = "longitudes of v component velocity" ;
9  double vlat(nz, nyp1, nx) ;
10 vlat:axis = "Y" ;
11 vlat:units = "degrees_north" ; vlat:long_name = "latitude" ;
12 vlat:comment = "latitudes of v component velocity" ;
13 double vlev(nz, nyp1, nx) ;
14 vlev:axis = "Z" ;
15 vlev:positive = "up" ;
16 vlev:units = "meters" ; vlev:long_name = "depth" ;
17 vlev:comment = "depths of v component velocity" ;
18 double w(time, nzp1, ny, nx) ;
19 w:units = "m/s" ;
20 w:long_name = "w component of the velocity" ;
21 w:coordinates = "wlon wlat wlev time" ;
22 double wlon(nzp1, ny, nx) ;
23 wlon:axis = "X" ;
24 wlon:units = "degrees_east" ; wlon:long_name = "longitude" ;
25 wlon:comment = "longitudes of w component velocity" ;
26 double wlat(nzp1, ny, nx) ;
27 wlat:axis = "Y" ;
28 wlat:units = "degrees_north" ;  wlat:long_name = "latitude" ;
29 wlat:comment = "latitudes of w component velocity" ;
30 double wlev(nzp1, ny, nx) ;
31 wlev:axis = "Z" ;
32 wlev:positive = "up" ;
33 wlev:units = "meters" ;    wlev:long_name = "depth" ;
34 wlev:comment = "depths of w component velocity" ;
```

ncdump gcom_restart.nc

```
double p(time, nz, ny, nx) ;
p:units = "bar" ;
p:long_name = "non-hydrostatic pressure" ;
p:coordinates = "lon lat lev time" ;
double T(time, nz, ny, nx) ;
T:units = "degrees celsius" ;
T:long_name = "temperature" ;
T:coordinates = "lon lat lev time" ;
double S(time, nz, ny, nx) ;
S:units = "psu" ;
S:long_name = "salinity" ;
S:coordinates = "lon lat lev time" ;
double lon(nz, ny, nx) ;
lon:axis = "X" ;
lon:units = "degrees_east" ;
lon:long_name = "longitude" ;
lon:comment = "longitude of grid center [p,S,T]" ;
double lat(nz, ny, nx) ;
lat:axis = "Y" ;
lat:units = "degrees_north" ;
lat:long_name = "latitude" ;
lat:comment = "latitude of grid center [p,S,T]" ;
double lev(nz, ny, nx) ;
lev:axis = "Z" ;
lev:positive = "up" ;
lev:units = "meters" ;
lev:long_name = "depth" ;
lev:comment = "depth of grid center [p,S,T]" ;
double D(time, nz, ny, nx) ;
D:units = "g/cm3" ;
D:long_name = "density" ;
D:coordinates = "lon lat lev time" ;
double Vortex_x(time, nzp1, nyp1, nxp1) ;
double Vortex_y(time, nzp1, nyp1, nxp1) ;
double Vortex_z(time, nzp1, nyp1, nxp1) ;
```

ncdump gcom_restart.nc

```
1
2  // global attributes:
3  :title = "OUTPUT␣File␣General␣Curvilinear␣Ocean␣Model␣GCOM" ;
4  :institution = "Computational␣Science␣Research␣Center␣(CSRC),␣San␣Diego␣State␣University" ;
5  data:
6
7   time = 12784.0002314815, 12784.0071759259, 12784.0141203704,
8      12784.0210648148, 12784.0280092593, 12784.0349537037, 12784.0418981481,
9      12784.0488425926, 12784.055787037, 12784.0627314815, 12784.0696759259,
10     12784.0766203704, 12784.0835648148, 12784.0905092593, 12784.0974537037,
11     12784.1043981481, 12784.1113425926, 12784.118287037, 12784.1252314815,
12     12784.1321759259, 12784.1391203704, 12784.1460648148, 12784.1530092593,
13     12784.1599537037, 12784.1668981481, 12784.1738425926, 12784.180787037,
14     12784.1877314815, 12784.1946759259, 12784.2016203704, 12784.2085648148,
15     12784.2155092593, 12784.2224537037, 12784.2293981481, 12784.2363425926,
16     12784.243287037, 12784.2502314815, 12784.2571759259, 12784.2641203704,
17     12784.2710648148, 12784.2780092593, 12784.2849537037, 12784.2918981481,
18     12784.2988425926, 12784.305787037, 12784.3127314815, 12784.3196759259,
19     12784.3266203704, 12784.3335648148 ;
20 }
```

Viewing NetCDF File Contents Using ncview