Advanced UNIX Lab Session

March 9 2010

Most of the reference material for this session is included in the relevant documents at:

http://www.earthsci.unimelb.edu.au/~kevin/UNIX_Course/

Advanced_UNIX_Lab_Session_2010.pdf
Introduction_to_Advanced_UNIX_2010.ppt
Intro_to_C-Shell_Programming_2008.pdf
Intro_to_Advanced_UNIX-Part_1_2008.pdf
Intro_to_Advanced_UNIX-Part_2_2008.pdf
Intro_to_Advanced_UNIX-Part_3_2008.pdf
zadv_unix.zip

This document includes extracts from the above plus some new material.

Introduction

The purpose of this tutorial is two-fold:

1. To highlight some basic features of C-shell scripting
2. To illustrate CMP-based utilities in conjunction with C-shell scripts

It is common to deal with meteorological variables e.g. mean sea level pressure, in gridded form e.g. on a global latitude-longitude 2.5 x 2.5° grid. The most common data formats for such variables are NetCDF and GRIB. You may be able to read these files directly with a software package e.g. NCL, GrADS, Matlab, as well as perform data processing e.g. compute the mean of a set of time slices (maps). Alternatively you can convert these formats to the 'conmap' (CMP) format (for NetCDF see: `read_nc2cmp`; for GRIB see: `wgrib/readgribn7` or `grb2nc`). It is probably easier to write your own Fortran programs to process the CMP files than to deal with the complexities of NetCDF or GRIB. There are utilities already available to process CMP files e.g. `splitcon`, `readcmp`, `statconmap6`, or for plotting purposes (`conmap7`). Output in CMP format can be converted to NetCDF (see: `cmp2cdl4`) for universal distribution. The choice or balance of data processing (CMP versus packages) depends on the nature of your research.

Software

Some useful software that is used in the Lab session is summarised here.

cmp2cdl4

This is a utility to translate a single of multi-map CMP (conmap) file into a NetCDF CDL text file. The CDL file may then be converted by `ncgen` into a binary NetCDF file for input to GrADS, Matlab, NCO or Panoply.

Usage: `cmp2cdl4 [-h] -n nmlist -i cmpfile -o cdlfile`

Documentation:
For a help screen: cmp2cdl4 –h

Namelist details:

```
 namelist /nmcdl/ var,
* attr_var_long_name, attr_var_units,
* gattr_desc, gattr_hist,
* date_time_fmt, date_time_type,
* nc_name, time_unlimited,
* map1, map2

Example namelist:
 &nmcdl
   var= 'H',
   date_time_type= 'YM',
   date_time_fmt= '(18x,I4,I2)',
   attr_var_long_name= 'H',
   attr_var_units= 'per Kelvin',
   gattr_desc= 'Monthly H (year-month) based
 on monthly HadISST T and NCEP
 Reanalysis E; Period Jan 1979 - Dec 2005',
   nc_name= 'my_H',
   map1= 5,
   map2= 10,
   time_unlimited=F,
   gattr_hist= 'Created by Kevin Keay',
 &end
```

Example: `cmp2cdl4 -n nmlist.txt -i cstatdat.cmp -o test.cdl`

`ncgen -b test.cdl` (uses name in CDL file)

`ncgen -o test.nc test.cdl`

conmap

This is a program based on NCAR Graphics for plotting gridded binary files in a simple format called 'conmap' (CMP, also known as CIF at CSIRO or the Bureau of Meteorology). There are a number of versions in use:

Solaris OS (atlas, orthus)

`conmap`: Original program
`conmap_kk`: An early enhanced version by Kevin Keay

Linux machines

`conmap`: Similar to `conmap_kk` on orthus
`conmap_kk`: An alias of `conmap`
`conmap5`: A later version (v. 5, 2005). Use this for vector plots (-V option)
`conmap7`: Most recent version (v.7.04, 2006). In general, use this one except for vector plots.

Usage: `conmap [options] cmpfile [< instruction_file]`
`conmap7[-k namelist_file][options] cmpfile [< ins_file]`

Documentation: http://www.earthsci.unimelb.edu.au/~kevin/conmap_man.pdf
This covers the main options but needs to be updated.
For brief help on conmap7:

`conmap7 --help` and:
`conmap7 --namelist`

to see some information about the namelist parameters (options `-k`, `-K`).

A remark on binary CMP (conmap) files

If you are using a Linux machine e.g. tide, you need to convert any Solaris CMP (conmap) files e.g. created on atlas, using a program written by Kevin Keay called `binswap`. (This is actually written in C!):

```
binswap -c pmsl.ncep.sun.cmp pmsl.ncep.linux.cmp
```

where `binswap` is run on the *Linux* machine.
The output CMP file (pmsl.ncep.linux.cmp) can now be read on a Linux machine.

The reverse procedure is also true: a Linux CMP file may be converted on a SUN with `binswap`. Note: If you create and use CMP files on a single platform e.g. Linux, then this operation is not necessary.

Answering prompts

Often you will use programs that require answers to questions as well as command-line arguments. A program used by many people in the group for plotting data on a latitude-longitude grid is called `conmap`, which is a Fortran program based on the free NCAR Graphics. There are several variations of this program e.g. `conmap7`, but the generic name `conmap` will be used for illustration. Here is an example.

At the prompt type:

```
conmap -SB pmsl.ncep.sun.cmp
```

You will be asked:
Do you want a grey scale rather than colour bands?
Answer n for no.
Enter format for key labels (6 chars) e.g. (f6.1)
Type (f6.1) (appropriate for pressure values)
Do you want to select your own colours? (y/n)
Answer n
Do you want the key?
Answer y
You should then see a message: DO NOT FORGET TO USE GLW_COLOR TO PRINT
A special graphics file called gmeta has been created.
At one stage the script glw_color was used for printing purposes. Today you can use:

```
g2ps -c gmeta
```

where –c means a colour plot.
This creates a Postcsript file called g.ps. You can use convert to change this to a graphics format e.g. PNG, JPEG, to insert into (say) a Microsoft Word document:

```
convert g.ps g.png
```

There are many options for `convert` – see: `man convert`

The Linux version is more recent than the SUN version – see for instance the `-trim` option.
You can view these graphics files with `gv` (Postscript) or `display` (general formats)
e.g. `gv g.ps`
`display g.png`

Now, rather than answering the prompts we can store our responses in a text file and control the program. Using:

```
nedit icon
```

Type:

```
n
(f6.1)
n
y
```

and save it. Then run the script:

```
#!/bin/csh -f
conmap -SB pmsl.ncep.sun.cmp < icon
g2ps -c gmeta
convert g.ps g.png
exit
```

The < means read the responses from the file called icon rather than the keyboard.
A variation on this theme is to include the responses in the script using the << operator:

```
#!/bin/csh -f
conmap -SB pmsl.ncep.sun.cmp << !      # The responses end at next instance of
!
n
(f6.1)
n
y
!
g2ps -c gmeta
convert g.ps g.png
exit
```

grb2nc

This a command based on the shell version of `xconv` (`convsh`). The basic usage is:

```
grb2nc -i gribfile -o ncfile
```

e.g. `grb2nc -i apr.2000.grb -o apr.2000.nc`

The converted file (apr.2000.nc) can be converted to CMP format using `read_nc2cmp`.

GrADS (`grads`)

GrADS is a free package that can be used to plot gridded data and also perform data processing. It has a scripting language so that complex procedures can be created. A useful feature is `sdfopen` which allows common NetCDF files to be read. There is a version for Windows and the Mac too. The current version is 1.9b4.

Usage: `grads`
Also see the documentation especially the tutorial.

Documentation: http://www.iges.org/grads/gadoc/index.html

`ncdump/ncgen`

These NetCDF utilites are useful for listing the contents of a NetCDF files (`ncdump`) and generating a NetCDF file from a CDL version of a CMP file (`cmp2cdl4` followed by `ncgen`). For `ncdump` the option `-h` gives a header dump (useful for setting parameters for read_nc2cmp) To list a variable use `-v` *varname* where *varname* is a valid variable in the NetCDF file.

NCL (`ncl`)

NCL (NCAR Command Language) is the successor to NCAR Graphics. It also has NetCDF manipulation utilities. Currently it is available only on the Solaris (SUN) machine atlas and the Linux machines abyss and vislab*. See the NCL website for documentation.

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

Using NCL in the School of Earth Sciences

The current version of NCL is 5.0.See:

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

The first point of call should be 'Getting Started With NCL':

http://www.ncl.ucar.edu/get_started.shtml

Some simple examples to try are at:

http://www.ncl.ucar.edu/Document/Manuals/Getting_Started/examples.shtml

On our machines you can access these examples (and others) like this:

```
ng4ex gsun01n
ng4ex gsun02n
...
ng4ex gsun11n
```

To run a NCL script e.g. gsun01n, use something like:

```
ncl gsun01n.ncl
```

A categorised list of NCL application examples is at:

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

`readcmp`

This is used in to list the contents of a single map CMP file or the first map of a concatenated CMP file.
For usage: `readcmp`

`readgribn7`

This is used in conjunction with `wgrib` to decode GRIB data files to CMP (conmap) format.
For usage: `readgribn7`
See Part3 for more information.

`read_nc2cmp`

This is used to decode common NetCDF data files, such as those available from reanalysis products, to CMP (conmap) format.
For usage: `read_nc2cmp`
See Part3 for more information.
Note: At this stage `read_nc2cmp` will **not** run on the VisLab (vislab*) machines since the utility `udunits` is unavailable. Try the RedHat machines e.g. cove, or the Solaris machine atlas
However, the output on the SUN machine (atlas) needs to be processed by `binswap` on a Linux machine to make it compatible with these machines.

wgrib

Usage: `wgrib [gribfile] [options]`

Documentation: http://www.cpc.ncep.noaa.gov/products/wesley/wgrib.html
A brief help screen is given by: `wgrib`
See Part 3 for more information.

Lab session – Advanced UNIX

Kevin Keay

March 8 2010

1. Log on to abyss:
   ```
   ssh -Y abyss
   ```
2. `cd ~`
   ```
   mkdir adv_unix
   cd adv_unix
   cp /home/keay/UNIX_Course/zadv_unix.zip .
   unzip zadv_unix.zip
   ```

   You will see these folders in `adv_unix`:

   ```
   data/  gmt/  lab/  reanal/
   ```

   The Lab session scripts are in `lab`, the decoding of a NetCDF file and data processing examples are in `reanal` and the GMT examples are in `gmt`.

   The scripts have names ending in .csh. When you write your own scripts you may call them whatever you wish e.g. `add_files`.
3. As a quick exposure to writing scripts:
   ```
   nedit s1.csh
   ```

   Type:

   ```
   #!/bin/csh -f
   set x = `whoami`   # Store the username in $x
   echo $x            # This will write the username on the screen
   ```

   Save it.
   At the shell prompt:

   ```
   csh -n s1.csh
   ```

This just checks for errors – the script is not executed.
Make the script an executable file: `chmod 755 s1.csh`
Run it:

```
s1.csh
```

You should see your username printed on the screen
e.g. kevin

4. Try out the other scripts: `s2.csh` – `s7.csh` or in UNIX, `s[2-7].csh` (!)
   To display the PNG files from s6.csh and s7.csh use: `display g.png`

Examples

s1.csh

```
#!/bin/csh -f
set x = `whoami`   # Store the username in $x
echo $x            # This will write the username on the screen
```

s2.csh

```
#!/bin/csh -f
set ff = `ls`      # $ff will contain all the filenames in the
current folder
@ nf = $#ff        # $nf is the number of filenames
@ i = 1
while ($i <= $nf)
  set f = ($ff[$i])
  echo "File "$i ":" $f      # Write each filename on screen
  @ i ++                     # Increments counter $i by 1
end
```

s3.csh

```
#!/bin/csh -f
foreach f (*.pal)
  echo "Input filename:" $f
  set o = `echo $f | sed -e "s/pal/txt/"`
  echo "Renaming $f to $o"
  mv $f $o
end
exit
```

Note: The *.pal files are renamed (moved) to *.txt. The original *.pal files are stored in zpal.zip. Before proceeding, restore these files:

```
unzip zpal.zip
```

s4.csh

```
#!/bin/csh -f
foreach f (pmsl.ncep.????????.cmp)
  echo "Input filename:" $f
  set d = `echo $f | cut -d. -f 3`    # d is date (field 3)
  echo "Date:" $d
  set yr =  `echo $d | cut -c 1-2`    # Use characters 1-2 as year
  set mn =  `echo $d | cut -c 3-4`    # Use characters 3-4 as
month
  set dy =  `echo $d | cut -c 5-6`    # Use characters 5-6 as day
  set hr =  `echo $d | cut -c 7-8`    # Use characters 5-6 as hour
  echo "Year: "$yr "Month: "$mn "Day: "$dy "Hour: "$hr
end
```

s5.csh

```
#!/bin/csh -f
echo "List 1"
ls pmsl.ncep.96060[1-3]*.cmp
echo "List 2"
ls pmsl.ncep.96060[1,3]*.cmp
echo "List 3"
ls pmsl.ncep.9606??{06,18}.cmp
```

s6.csh

```
#!/bin/csh -f
if ($#argv == 0) then
  echo "Usage: s6.csh CMP_file"
  exit
else
  set infile = ($1)   # Assign argument 1 to infile
  echo "CMP file: "$infile  # or ${infile}
endif

# Create plot (gmeta) using instruction file icon (answers to
prompts)
conmap7 -k icon_k.band.s6 -SB $1 < icon
# Convert gmeta to Postscript (g.ps)
g2ps -c gmeta
# Convert g.ps to g.png (PNG file)
convert -trim -density 100 g.ps gs6.png

echo "Plot file: gs6.png"
# Next line - needs to have \! not just ! - confuses csh
echo "Done\!"

exit
```

Note: The script `s6.csh` uses additional files (icon_k.band.s6 and icon) to work correctly. These files contain parameters and responses for `conmap7`. In addition you may use any valid pressure CMP file as an input.

s7.csh

```
#!/bin/csh -f

# In this example it is ASSUMED that you will give a conmap
filename
# This becomes argument 1 ($1)

if ($#argv == 1) then
echo "Filename is "$1
conmap -SB $1 << !     # The responses end at next instance of !
n
(f6.1)
n
y
!
g2ps -c gmeta
convert g.ps g.png

else
  echo "ERROR: Give a filename, silly\!"  # Note \! not just !
endif

exit
```

Note: The script `s7.csh` uses the `<< ! … !` structure to give the responses for `conmap7`.

NCL examples

There are some introductory examples that you can try out using the `ng4ex` command:

```
ng4ex gsun01n
ng4ex gsun02n
...
ng4ex gsun11n
```

To run a NCL script e.g. gsun01n, use something like:

```
ncl gsun01n.ncl
```

Data processing

1. An example of decoding a NetCDF file and data processing

Note: The C-shell scripts and `conmap7/ausmap` instruction files are in **adv_unix/reanal**.

The required NetCDF files are in **adv_unix/data**.

Ensure that you run these examples from: **adv_unix/reanal**

Assume we have downloaded a NetCDF file called pmsl_198007_part_ncep2.nc from the NCEP Reanalysis (NCEP2) web site. It is located in folder adv_unix/data. A header dump i.e. `ncdump -h pmsl_198007_part_ncep2.nc`, indicates that the pressure variable is named mslp and has units of Pa (we will rescale to hPa). There are 16 maps from July 1 1980 OOUTC - July 4 1980 18UTC.

Hence we may decode all of the maps in the NetCDF file with the command:

```
read_nc2cmp -i ../data/pmsl_198007_part_ncep2.nc -o pmsl_ncep2.cmp \
-u mslp -r NCEP2 -v PMSL -s 0.01 -p /home/keay/bin/udunits.dat
```

Note: \ is the C-shell continuation character in scripts – omit \ and type as one line interactively.

The multi-map CMP (conmap) file is named pmsl_ncep2.cmp and contains 16 maps.

We may obtain the individual maps as separate files with:

```
splitcon -n -l pmsl_ncep2.cmp
```

The CMP files are: pmsl.ncep2.1980070100.cmp - pmsl.ncep2.1980070418.cmp

We can take the average of these 16 maps with:

```
statconmap pmsl.ncep2.198007*.cmp 1 ave.cmp
```

The '1' indicates that the output CMP file (ave.cmp) is an average file.

For more general purposes (and larger grid sizes) use: `statconmap6`:

```
statconmap6  pmsl.ncep2.198007*.cmp {ave,sd,var,cnt}.cmp
```

In this case you need to specify four output files (average, standard deviation, variance and count of non-missing values); there is no '1'. Type: `statconmap6` for usage.

You may list the contents of the file ave.cmp using `readcmp`:

```
readcmp ave.cmp >! List.txt
```

`nedit list.txt` (this will allow you to view the contents of list.txt)

We might want to express a given map as an anomaly from this average (usually we would use a longer averaging period). This can be achieved by:

```
conmanip2 -s pmsl.ncep2.1980070218.cmp ave.cmp diff.cmp
```

The option `-s` is 'subtract' - for usage: `conmanip2`

In this example diff.cmp is the anomaly for July 2 1980 18UTC

i.e. (map at this time) - (average of 16 maps).

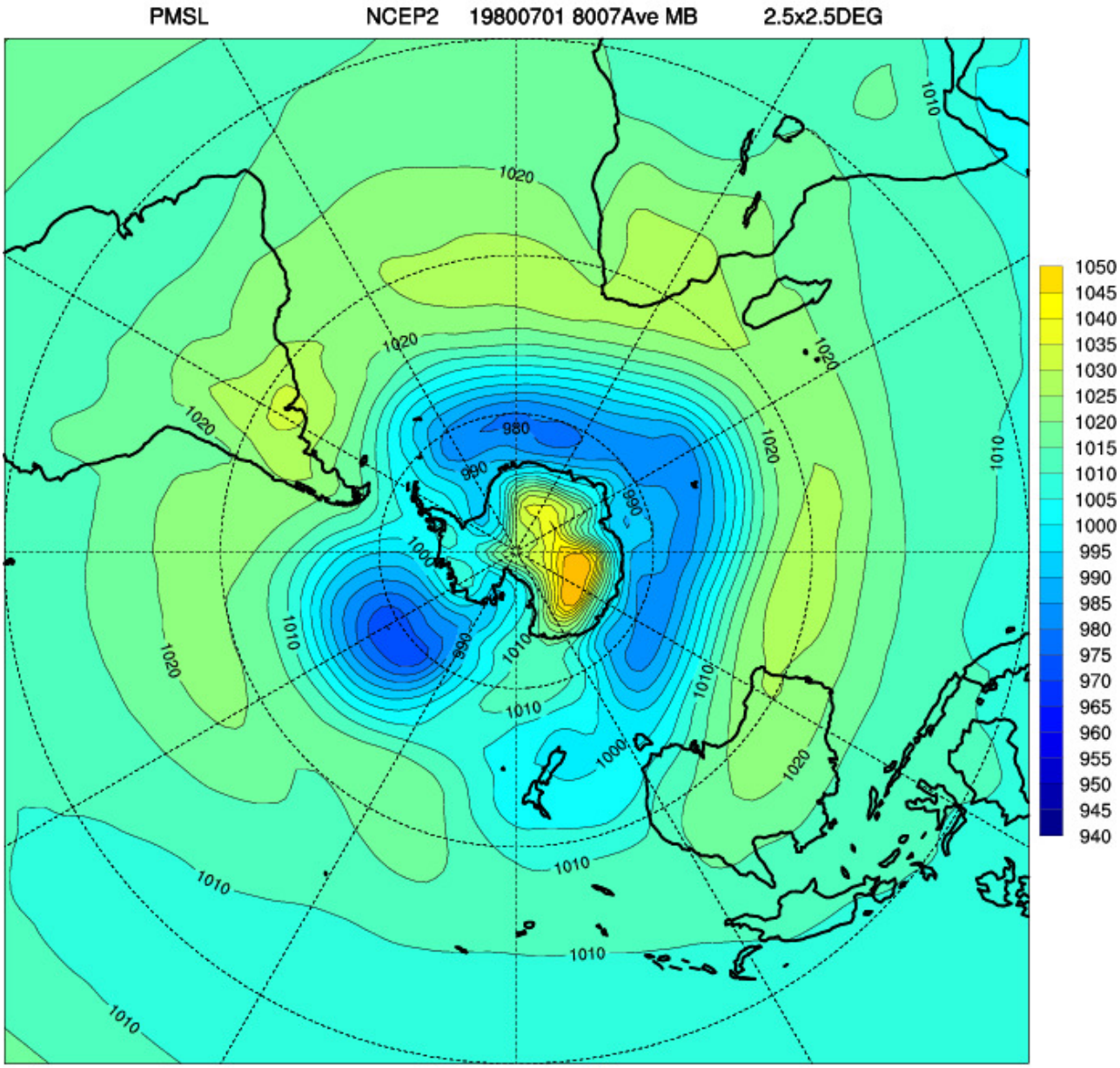
We can produce a southern hemisphere plot of the average map with:

```
plot_ave.csh ave.cmp
```

You may display the plot with: `display gave.png`

(The PNG file name is printed on the screen during script execution).

Average



Note: For polar stereographic plots, there is a plotting glitch if lon 0 is not duplicated as lon 360. The program `fixcon` (for usage: `fixcon`) will duplicate the lon 0 at lon 360.

```
fixcon -d diff.cmp diff.cmp
```

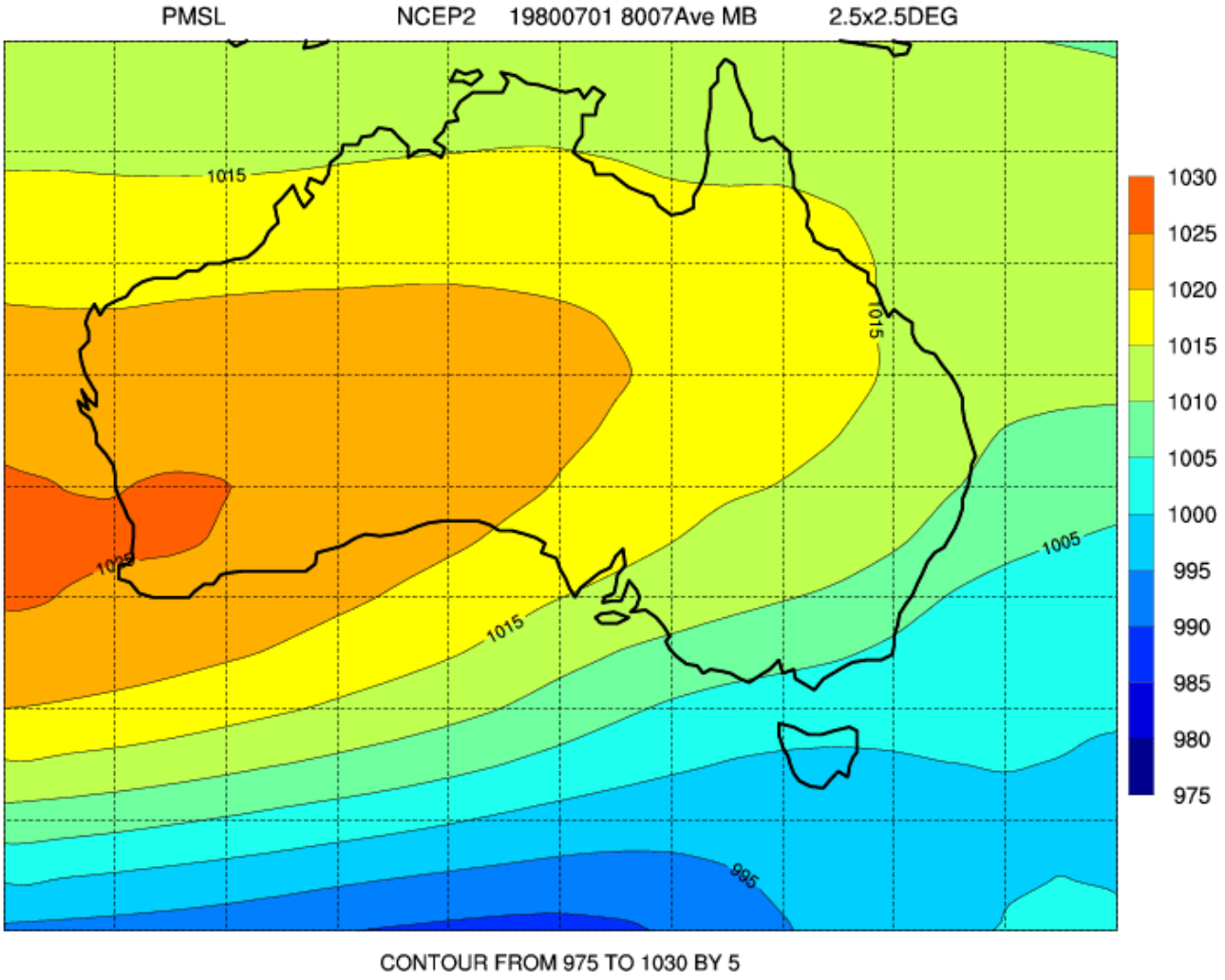
```
fixcon -d ave.cmp ave.cmp
```

You may plot the file diff.cmp with: `plot_diff.csh diff.cmp`

Similarly a plot over the Australian region is created with:

```
plot_ave.aus.csh ave.cmp
```

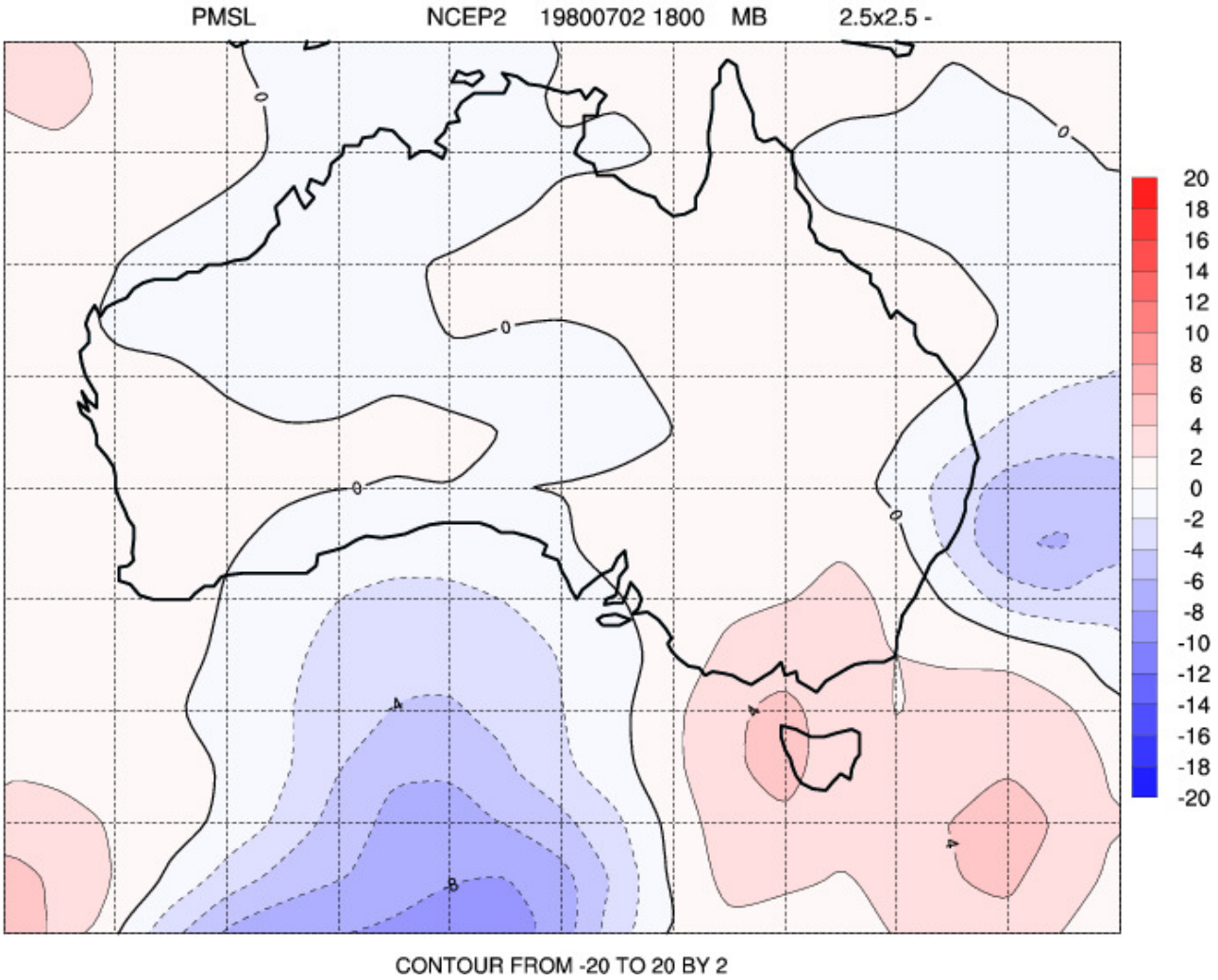
Average over Australian region



A plot of the anomaly (difference) map for July 2 1980 18UTC is produced by:

```
plot_diff.aus.csh diff.cmp
```

Anomaly over Australia



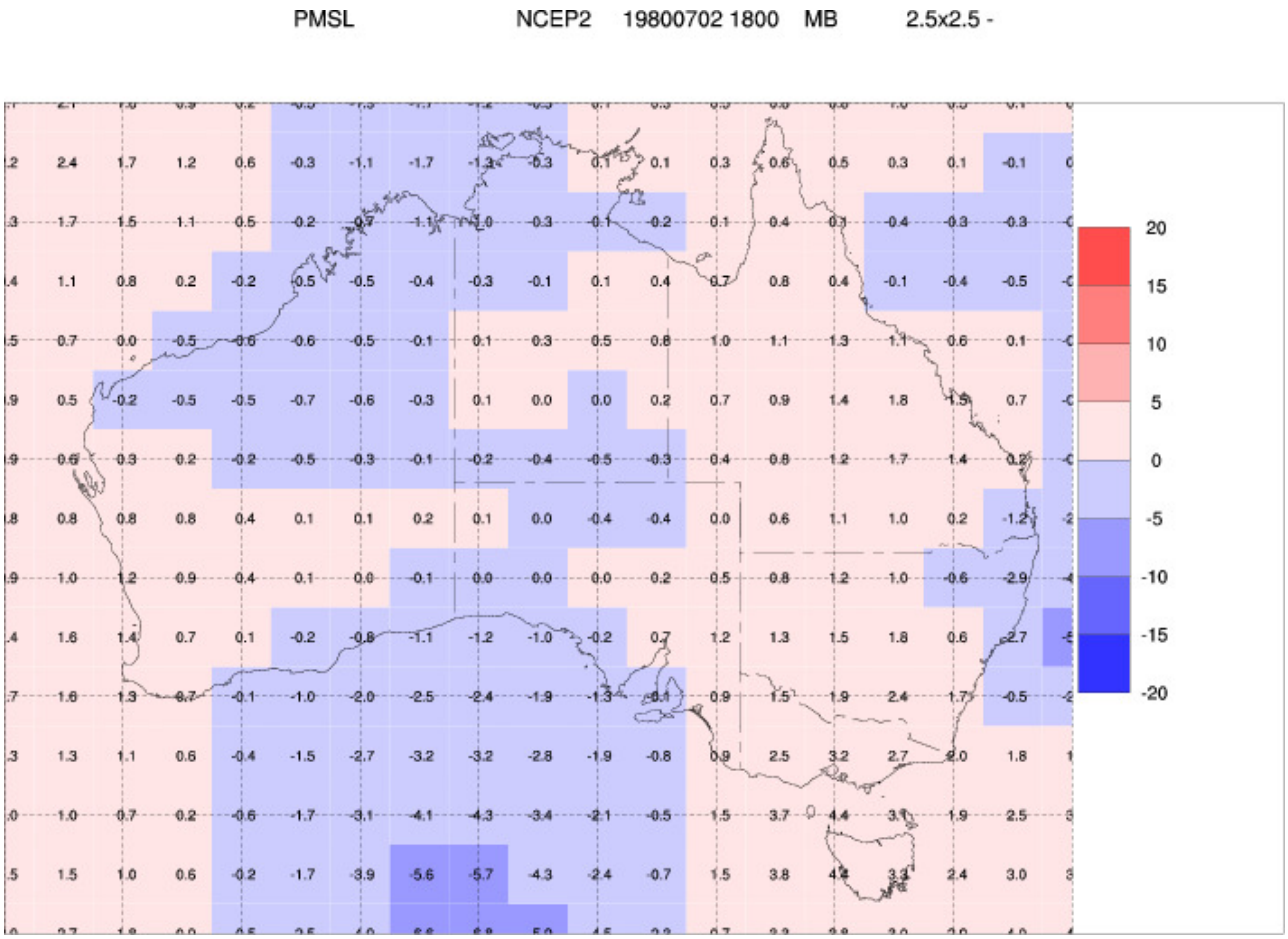
A plot of the actual gridpoint values of the anomalies for this particular time may be created with `ausmap`, using the raster (`-X`) and print (`-P`) options:

```
ausmap -X -P -2 -s diff.cmp < iaus.raster.diff

g2ps -c gmeta

convert -trim -density 100 g.ps gdiff_raster.png

display gdiff_raster.png
```



Perhaps we want to use the anomaly map with a package like GrADS.

We can convert the CMP file (diff.cmp) to a NetCDF file by:

```
cmp2cdl4 -n nmlist.txt -i diff.cmp -o j.cdl
```

```
ncgen -o diff.nc j.cdl
```

See the namelist file (nmlist.txt) for appropriate NetCDF attribute

values. These need to be changed depending on the variable and data set. `cmp2cdl4` is a recently written program so the documentation is sparse.

The NetCDF file (diff.nc) can read by GrADS:

```
grads
```

(a graphics window will open)

At the grads prompt (ga->) open the NetCDF file (diff.nc):

```
sdfopen diff.nc
```

To check the file information:

```
q file
```

This gives a listing:

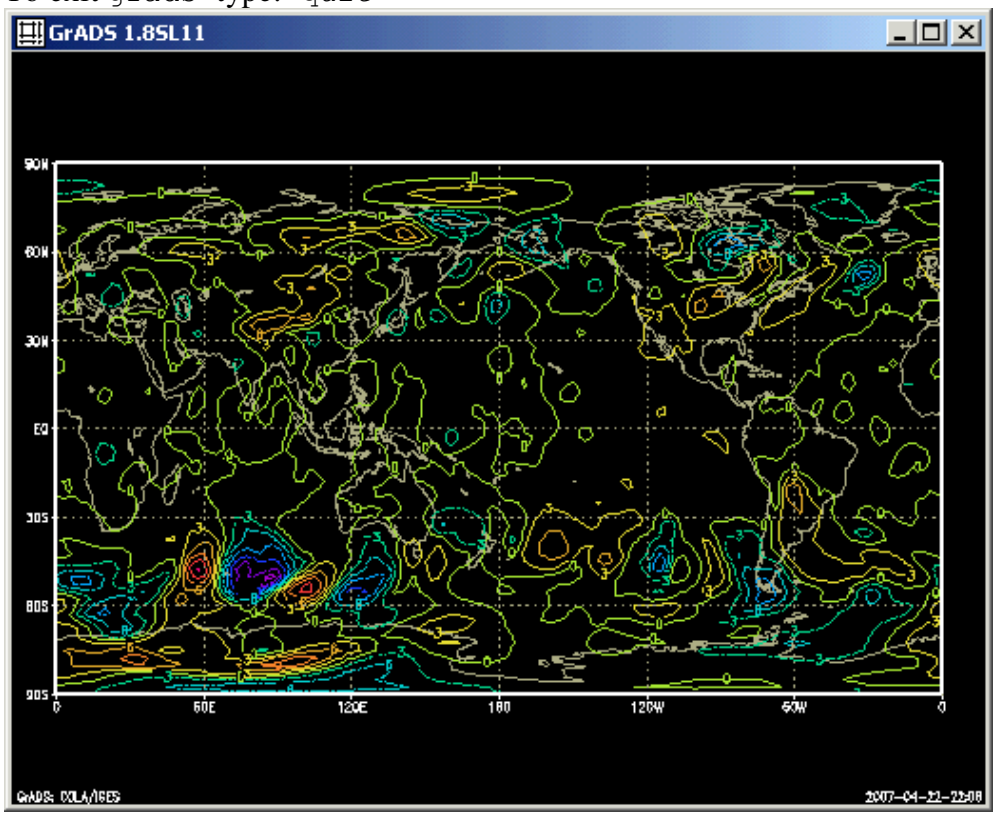
```
File 1 :
  Descriptor: diff.nc
  Binary: diff.nc
  Type = Gridded
  Xsize = 144  Ysize = 73  Zsize = 1  Tsize = 1
  Number of Variables = 1
    diff_mslp 0 -999 diff_MSLP
```

The GrADS variable is named diff_mslp (diff_MSLP is the long NetCDF name).

To display the single map:

```
d diff_mslp
```

To exit `grads` type: `quit`



2. Decoding reanalysis data

Decoding NetCDF data to CMP ('conmap') format

Note: This section is for reference only.
The program `read_nc2cmp` can handle common NetCDF files, especially the reanalysis products.

Usage: For brief help: `read_nc2cmp`
and for some examples: `read_nc2cmp -help`

```
read_nc2cmp: Version 2.3 (Jun 4 2008)

 Usage: read_nc2cmp [--help][-D idbg][-i ncfile][-o cmpfile][-d
"lon,lat,time"]

 [-u uservar][-g gridtype][-l levelvar][-L ilev][-r rtype][-s uscal][-
v vtype]

 [-U units][-m no_maps][-p udunits][-M "map1,map2"][-T "ttype"]


 D: 0= None 1= Basic 2= Verbose 3= Print dimension arrays to file
fort.10

 T: ttype: CDO, fcst

    fcst: Adds 6 hr to date-time; use with NCEP/NCEP2 10m winds

 Note: Max. sizes of text variables

 gridtype: 10 rtype: 5 vtype: 8 units: 8

   --help: Gives some examples
```

Note: If your data file is a GRIB file then you can use `readgribn7` in conjunction with `wgrib` to convert the data to CMP. An easier way is convert the GRIB file to NetCDF using `grb2nc`

e.g. `grb2nc -i test.grb -o test.nc`

Then convert the NetCDF file to CMP using `read_nc2cmp`. Furthermore the NetrCDF file can be directly imported into packages like `GrADS` and NCL.

Some examples

Many of the default settings of the options will be correct for common reanalysis products. The `-d` option is set for NCEP and NCEP2 by default, assuming that the longitude, latitude and time

variables are named `lon`, `lat` and `time`.

If you are unsure, use `ncdump` e.g. `ncdump -h hgt.1980.nc`, to check the NetCDF file header for the names of dimensions and variables – these are case-sensitive e.g. SLP is not the same as slp.

(1) ERA40 with no level variable

The output from `ncdump` for hgt.2002.Jun.500hPa.nc in folder ncdata is:

```
netcdf hgt.2002.Jun.500hPa {
dimensions:
        longitude = 144 ;
        latitude = 73 ;
        time = UNLIMITED ; // (120 currently)
variables:
        float longitude(longitude) ;
                longitude:units = "degrees_east" ;
                longitude:long_name = "longitude" ;
        float latitude(latitude) ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "latitude" ;
        int time(time) ;
                time:units = "hours since 1900-01-01 00:00:0.0" ;
                time:long_name = "time" ;
        short z(time, latitude, longitude) ;
                z:scale_factor = 0.217434325978148 ;
                z:add_offset = 51602.802096924 ;
                z:_FillValue = -32767s ;
                z:missing_value = -32767s ;
                z:units = "m**2 s**-2" ;
                z:long_name = "Geopotential" ;

// global attributes:
                :Conventions = "CF-1.0" ;
                :history = "2006-05-11 03:18:51 GMT by mars2netcdf-0.92" ;
}
```

Hence:

```
read_nc2cmp -i ncdata/hgt.2002.Jun.500hPa.nc -o j.cmp \
-d "longitude,latitude,time" -u z -D 3 -r ERA40 -m 2 \
-p /home/keay/bin/udunits.dat
```

The user variable is geopotential (`z`) and the three basic dimensions are named `longitude`, `latitude` and `time`.

The dataset is ERA40 and as a test we will output the first two maps. Omit `-m` option to get all maps.

`-D` is the debug option; 3 prints the dimensions to a file called fort.10. You can omit `-D` option.

Note: For ERA40 geopotential, the program will divide by g= 9.807 m s**-2 to give geopotential height (m).

(2) ERA40 with a level variable

The output from `ncdump` for hgt.200208.nc in folder ncdata is:

```
netcdf hgt.200208 {
dimensions:
        longitude = 144 ;
        latitude = 73 ;
        levelist = 23 ;
        time = UNLIMITED ; // (62 currently)
variables:
        float longitude(longitude) ;
                longitude:units = "degrees_east" ;
                longitude:long_name = "longitude" ;
        float latitude(latitude) ;
                latitude:units = "degrees_north" ;
                latitude:long_name = "latitude" ;
        int levelist(levelist) ;
                levelist:units = "millibars" ;
                levelist:long_name = "pressure_level" ;
        int time(time) ;
                time:units = "hours since 1900-01-01 00:00:0.0" ;
                time:long_name = "time" ;
        short z(time, levelist, latitude, longitude) ;
                z:scale_factor = 7.53787087081502 ;
                z:add_offset = 241940.397676004 ;
```

```
            z:_FillValue = -32767s ;
            z:missing_value = -32767s ;
            z:units = "m**2 s**-2" ;
            z:long_name = "Geopotential" ;

// global attributes:
            :Conventions = "CF-1.0" ;
            :history = "2006-05-08 06:31:19 GMT by mars2netcdf-0.92" ;
}
```

There are 23 levels. If you use `ncdump -v levelist` then you can see the levels:

```
 levelist = 1, 2, 3, 5, 7, 10, 20, 30, 50, 70, 100, 150, 200, 250, 300, 400,
    500, 600, 700, 775, 850, 925, 1000 ;
```

If you want the 500 hPa level then you require `levelist(17)`.

Hence:

```
read_nc2cmp -i ncdata/hgt.200208.nc -o j.cmp
 -d "longitude,latitude,time" -u z -D 3 -r ERA40 -l levelist -L 17 -m 2 \
 -p /home/keay/bin/udunits.dat
```

(3) NCEP2 with no level variable

The output from `ncdump` for hgt.58.0500.nc in the current folder is:

```
netcdf hgt.58.0500 {
dimensions:
       time = 1460 ;
       lat = 73 ;
       lon = 144 ;
variables:
       double time(time) ;
              time:units = "hours since 1-1-1 00:00:0.0" ;
              time:long_name = "Time" ;
              time:actual_range = 17154744., 17163498. ;
              time:delta_t = "0000-00-00 06:00:00" ;
       float lat(lat) ;
```

```
            lat:units = "degrees_north" ;
            lat:actual_range = 90.f, -90.f ;
            lat:long_name = "Latitude" ;
      float lon(lon) ;
            lon:units = "degrees_east" ;
            lon:long_name = "Longitude" ;
            lon:actual_range = 0.f, 357.5f ;
      short hgt(time, lat, lon) ;
            hgt:long_name = "4xDaily Geopotential height" ;
            hgt:actual_range = -513.f, 32308.f ;
            hgt:valid_range = -700.f, 35000.f ;
            hgt:units = "m" ;
            hgt:add_offset = 32066.f ;
            hgt:scale_factor = 1.f ;
            hgt:missing_value = 32766s ;
            hgt:precision = 0s ;
            hgt:least_significant_digit = 0s ;
            hgt:GRIB_id = 7s ;
            hgt:GRIB_name = "HGT" ;
            hgt:var_desc = "Geopotential height\n",
    "H" ;
            hgt:dataset = "NMC Reanalysis\n",
    "L" ;
            hgt:level_desc = "Multiple levels\n",
    "F" ;
            hgt:statistic = "Individual Obs\n",
    "I" ;
            hgt:parent_stat = "Other\n",
    "-" ;
}
```

Hence:

```
read_nc2cmp -i hgt.58.0500.nc -o j.cmp -d "lon,lat,time" -u hgt \
 -D 2 -r NCEP2 -m 2 -p /home/keay/bin/udunits.dat
```

Note that the variable is called `hgt` and the dataset is NCEP2. The `-D` option with 2

gives some extra information. Leave out the `-m` option to give all maps.

For NCEP2 or NCEP data you probably just need to change the `-u` option (and `-v` and `-U` options for variables other than geopotential height). See (4) below.
Look at the `ncdump` of the file (`ncdump -h yourfile.nc`) and check. In the above example, under Variables:

```
short hgt(time, lat, lon) o
```

Hence the variable is `hgt` => `-u hgt`

(4) NCEP with a level variable

The variable is specific humidity.

```
netcdf shum.2005.500 {
dimensions:
       lon = 144 ;
       lat = 73 ;
       level = 1 ;
       time = UNLIMITED ; // (1460 currently)
variables:
       float level(level) ;
              level:units = "millibar" ;
              level:actual_range = 500.f, 500.f ;
              level:long_name = "Level" ;
              level:positive = "down" ;
              level:GRIB_id = 100s ;
              level:GRIB_name = "hPa" ;
       float lat(lat) ;
              lat:units = "degrees_north" ;
              lat:actual_range = 90.f, -90.f ;
              lat:long_name = "Latitude" ;
       float lon(lon) ;
              lon:units = "degrees_east" ;
              lon:long_name = "Longitude" ;
              lon:actual_range = 0.f, 357.5f ;
```

```
       double time(time) ;
              time:units = "hours since 1-1-1 00:00:0.0" ;
              time:long_name = "Time" ;
              time:actual_range = 17566752., 17575506. ;
              time:delta_t = "0000-00-00 06:00:00" ;
       short shum(time, level, lat, lon) ;
              shum:long_name = "4xDaily specific humidity" ;
              shum:valid_range = -1.e-04f, 0.06543f ;
              shum:actual_range = 0.f, 0.009062f ;
              shum:units = "kg/kg" ;
              shum:add_offset = 0.032666f ;
              shum:scale_factor = 1.e-06f ;
              shum:missing_value = 32766s ;
              shum:precision = 6s ;
              shum:least_significant_digit = 5s ;
              shum:GRIB_id = 51s ;
              shum:GRIB_name = "SPFH" ;
              shum:var_desc = "Specific humidity\n",
    "Q" ;
              shum:dataset = "NMC Reanalysis\n",
    "L" ;
              shum:level_desc = "Multiple levels\n",
    "F" ;
              shum:statistic = "Individual Obs\n",
    "I" ;
              shum:parent_stat = "Other\n",
    "-" ;


// global attributes:
              :Conventions = "COARDS" ;
              :title = "4x daily NMC reanalysis (2005)" ;
              :history = "Wed May 31 18:13:10 2006: /usr/local/bin/ncrcat -O -d
level,500.000000 -d lat,-90.000000,90.000000 -d lon,0.000000,357.500000 -d time,0,1459
/Datasets/ncep.reanalysis/pressure/shum.2005.nc
/Public/www/128.250.120.93.150.18.13.8.nc\n",
    "created 2005/01/03 by Hoop (netCDF2.3)" ;
              :description = "Data is from NCEP initialized reanalysis\n",
    "(4x/day).  It consists of most variables interpolated to\n",
    "pressure surfaces from model (sigma) surfaces." ;
```

```
            :platform = "Model" ;
}
```

Based on the above NetCDF header dump the following command will create a concatenated (multi-map) conmap file with a useful header for each map:

```
read_nc2cmp -i shum.2005.500.nc -o j.cmp -d "lon,lat,time" -u shum -D 2 \
 -r NCEP -m 2 -l level -L 1 -v SHUM500 -U "'kg/kg'" \
 -p /home/keay/bin/udunits.dat
```

We set the variable name in the conmap header to be `SHUM500 (-v)` and the units to be `kg/kg (-U)` (note the extra single quotes to ensure that the `/` is treated as text).

(5) NCEP Mean sea level pressure

Consider the header dump of the NetCDF file slp.2004.nc i.e. `ncdump -h slp.2004.nc`

```
netcdf slp.2004 {
dimensions:
       lon = 144 ;
       lat = 73 ;
       time = UNLIMITED ; // (1464 currently)
variables:
       float lat(lat) ;
              lat:units = "degrees_north" ;
              lat:actual_range = 90.f, -90.f ;
              lat:long_name = "Latitude" ;
       float lon(lon) ;
              lon:units = "degrees_east" ;
              lon:long_name = "Longitude" ;
              lon:actual_range = 0.f, 357.5f ;
       double time(time) ;
              time:units = "hours since 1-1-1 00:00:0.0" ;
              time:long_name = "Time" ;
              time:actual_range = 17557968., 17566746. ;
              time:delta_t = "0000-00-00 06:00:00" ;
       short slp(time, lat, lon) ;
              slp:long_name = "4xDaily Sea Level Pressure" ;
              slp:valid_range = 87000.f, 115000.f ;
```

```
            slp:actual_range = 92700.f, 111370.f ;
            slp:units = "Pascals" ;
            slp:add_offset = 119765.f ;
            slp:scale_factor = 1.f ;
            slp:missing_value = 32766s ;
            slp:precision = 0s ;
            slp:least_significant_digit = -1s ;
            slp:GRIB_id = 2s ;
            slp:GRIB_name = "PRMSL" ;
            slp:var_desc = "Sea Level Pressure\n",
    "P" ;
            slp:dataset = "NMC Reanalysis\n",
    "L" ;
            slp:level_desc = "Sea Level\n",
    "I" ;
            slp:statistic = "Individual Obs\n",
    "I" ;
            slp:parent_stat = "Other\n",
    "-" ;

// global attributes:
            :Conventions = "COARDS" ;
            :title = "4x daily NMC reanalysis (2004)" ;
            :base_date = 2004s, 1s, 1s ;
            :history = "created 2004/01/03 by Hoop (netCDF2.3)" ;
            :description = "Data is from NMC initialized reanalysis\n",
    "(4x/day).  It consists of most variables interpolated to\n",
    "pressure surfaces from model (sigma) surfaces." ;
            :platform = "Model" ;
}
```

(1) To decode maps 5-8 of this mean sea level pressure file use the following command:

```
read_nc2cmp -i slp.2004.nc -o jj.cmp -u slp -r NCEP -v PMSL -s 0.01 -M "5,8" \
-p /home/keay/bin/udunits.dat
```

The pressure variable is named `slp` (`-u` option).

We need to scale the pressure in Pa to hPa i.e. apply a scaler of 0.01 (`-s` option).

The `-r` and `-v` options are for setting the conmap header for the cyclone tracking scheme but may be used for general purposes. The `-M` option gives the map range to be decoded i.e. maps 5-8.

The screen output during program execution is:

```
 NOTE: User scaler:   0.00999999978
 Output map range:  5 -  8
 NetCDF file opened successfully (ncid= 3)
 Inquiring about variables ...
 Reading longitudes ...
 Reading latitudes ...
 Reading times ...
 Reading attributes ...
 No. of maps to be extracted:  4
 Reading user variable ...
     5:PMSL                          NCEP      20040102 0000    MB
2.5x2.5DEG
     6:PMSL                          NCEP      20040102 0600    MB
2.5x2.5DEG
     7:PMSL                          NCEP      20040102 1200    MB
2.5x2.5DEG
     8:PMSL                          NCEP      20040102 1800    MB
2.5x2.5DEG
 NetCDF file closed successfully (ncid= 3)
 Output conmap file: jj.cmp
 Finished!
```

The file jj.cmp contains the four decoded maps.

(2) The first 10 maps may be decoded with:

```
read_nc2cmp -i slp.2004.nc -o jj.cmp -u slp -r NCEP -v PMSL -s 0.01 -m 10 \
-p /home/keay/bin/udunits.dat
```

(3) The entire file (1464 maps) may be decoded with:

```
read_nc2cmp -i slp.2004.nc -o pmsl.2004.cmp -u slp -r NCEP -v PMSL -s 0.01 \
```

```
-p /home/keay/bin/udunits.dat
```