Using GRIB Tools

Computer User Training Course 2016

Paul Dando & Carsten Maass

User Support

advisory@ecmwf.int

ECMWF

Contents

ecCodes command line tools – basic concepts

- The ecCodes tools are a set of command line programs for interactive and batch processing of GRIB and BUFR data
- They provide ready and tested solutions to the most common processing of GRIB and BUFR data
- Their use will avoid the need to write new code and thus speed up your work
  - Consider using ecCodes tools instead of writing your own program
- The tools are provided with a common set of options so that it is quick to apply the same options to different tools
- Use of the tools is recommended whenever possible!

Tools have been created in an attempt to provide easy to use command line solutions for the most common GRIB handling and coding/decoding demands.

Generic ecCodes tools

- There is a tool for getting information about the ecCodes installation
  - codes_info
- There is a tool for counting GRIB and BUFR messages
  - codes_count

grib_count counts grib messages only

codes_count counts both GRIB and BUFR (and metar and GTS and netcdf) messages

GRIB Tools – basics

All of the tools use a common syntax

grib_<tool> [options] grib_file [grib_file] … [output_grib]

- **Tool to count the messages in a GRIB file**
  - grib_count
- **Tools to inspect the content of and compare GRIB files**
  - grib_ls, grib_dump, grib_get, grib_get_data, grib_compare
- **Tool to copy some messages**
  - grib_copy
- **Tools to change the content of a GRIB message**
  - grib_set, grib_filter
- **Tool to convert a GRIB file to netCDF format**
  - grib_to_netcdf

Getting help

- **UNIX ‘man’-style pages are available for each tool by running the tool without any options or input file**

```
> grib_dump
NAME    grib_dump

DESCRIPTION
        Dump the content of a grib file in different formats.

USAGE
        grib_dump [options] grib_file grib_file ...

OPTIONS
        -O      Octet mode. WMO documentation style dump.
        -D      Debug mode.
        -d      Print all data values.
...
```

Documentation

- **The ecCodes manual is available at**

  https://software.ecmwf.int/wiki/display/ECC/ecCodes+Home

- **The GRIB Tools are documented at**

  https://software.ecmwf.int/wiki/display/ECC/GRIB+tools

  Includes some examples of how to use the tools

- **The WMO FM 92 GRIB Edition 1 and GRIB Edition 2 Manuals can be obtained from**

  http://www.wmo.int/pages/prog/www/WMOCodes.html

- **The ecCodes software can be downloaded from**

  https://software.ecmwf.int/wiki/display/ECC/Releases

codes_info – information about ecCodes installation

The codes_info tool gives basic information about the ecCodes package being used

- ecCodes Version
- Path to definition files: ECCODES_DEFINITION_PATH
- Path to sample files: ECCODES_SAMPLES_PATH

```
> codes_info
eccodes Version 0.13.0

Default definition files path is used: /usr/local/apps/eccodes/0.13.0/
   share/eccodes/definitions
Definition files path can be changed setting ECCODES_DEFINITION_PATH
   environment variable

Default SAMPLES path is used: /usr/local/apps/eccodes/0.13.0/share/
   eccodes/samples
SAMPLES path can be changed setting ECCODES_SAMPLES_PATH environment
   variable
```


Definition Path points to files describing the code standards.

Samples Path points to template files.

GRIB keys

- **For definitions of edition independent keys, GRIB1 or GRIB2 keys see**

  http://apps.ecmwf.int/codes/grib/

- **Usage of edition independent keys should be preferred**

Keys are keywords giving access to the content of GRIB messages.

grib_count – count GRIB messages

- **Counts (very quickly) the number of GRIB messages in a list of files**
- **Syntax**

```
grib_count grib_file1 [grib_file2 …]
```

(takes wildcards)

- Only option is –v for verbose output
- Is very fast
- Doesn't count BUFR messages (any longer)

/scratch/ectrain/trx/grib_api_data/grib_api_timing.grib

grib_dump – dump content of GRIB files

- **Use grib_dump to dump the content of a file containing one or more GRIB messages**
- **Various output formats are supported**
  - **Octet mode provides a WMO documentation style dump**
  - **Debug mode prints all keys available in the GRIB file**
  - **Octet and Debug modes cannot be used together**
  - **Octet content can also be printed in hexadecimal format**
- **Options also exist to print key aliases and key type information**
- **Output to JSON (JavaScript Object Notation)**
  - **Easy to process**

Basically 2 mutually exclusive output modes area available:

Octet mode	follows the WMO standard
Debug mode	prints all available keys, including computed

JSON or JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

grib_dump – usage

```
grib_dump [options] grib_file grib_file …
```

Basic options

`-O`	**Octet mode (WMO documentation style)**
`-D`	**Debug mode**
`-a`	**Print key alias information**
`-t`	**Print key type information**
`-H`	**Octet content in Hexadecimal**
`-D`	**Debug mode**
`-w key[:{s|i|d}]{=|!=}value,…`	**Where clause**
`-j`	**JSON output**
`-V`	**Print ecCodes Version**
`…`	

grib_dump

NAME grib_dump

DESCRIPTION
Dump the content of a grib file in different formats.

USAGE
grib_dump [options] grib_file grib_file ...

OPTIONS
-O Octet mode. WMO documentation style dump.
-D Debug mode.
-d Print all data values.
-j JSON mode (JavaScript Object Notation).
-C C code mode. A C code program generating the message is dumped.
-t Print type information.
-H Print octet content in hexadecimal format.
-a Dump aliases.
-w key[:{s/d/i}]{=/!=}value,key[:{s/d/i}]{=/!=}value,...
Where clause.
Messages are processed only if they match all the key/value constraints.
A valid constraint is of type key=value or key!=value.
For each key a string (key:s), a double (key:d) or an integer (key:i)
type can be specified. Default type is string.
-s key[:{s/d/i}]=value,key[:{s/d/i}]=value,...
Key/values to set.
For each key a string (key:s), a double (key:d) or an integer (key:i)
type can be defined. By default the native type is set.
-M Multi-field support off. Turn off support for multiple fields in single grib message.
-T T | B | M | A Message type. T->GTS, B->BUFR, M->METAR (Experimental),A->Any (Experimental).
The input file is interpreted according to the message type.
-7 Does not fail when the message has wrong length
-V Version.
-X offset
Input file offset in bytes. Processing of the input file will start from "offset".
-x Fast parsing option, only headers are loaded.

JSON or JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.
It is used primarily to transmit data between a server and web application, as an alternative to XML.

grib_dump – examples

```
> grib_dump file.grib1
***** FILE: file.grib1
#==============   MESSAGE 1 ( length=3280398 )             ==============
GRIB {
  editionNumber = 1;
  table2Version = 128;
  # European Center for Medium-Range Weather Forecasts (grib1/0.table)
  centre = 98;
  generatingProcessIdentifier = 139;
  # Geopotential  (m**2 s**-2)  (grib1/2.98.128.table)
  indicatorOfParameter = 129;
  # Isobaric level pressure in hectoPascals  (hPa)  (grib1/3.table)
  indicatorOfTypeOfLevel = 100;
  level = 1000;
  # Forecast product valid at reference time + P1  (P1>0)  (grib1/5.table)
  timeRangeIndicator = 0;
  # Unknown code table entry (grib1/0.ecmf.table)
  subCentre = 0;
  paramId = 129;
  #-READ ONLY- units = m**2 s**-2;
  #-READ ONLY- nameECMF = Geopotential;
  #-READ ONLY- name = Geopotential;
  decimalScaleFactor = 0;
  dataDate = 20110223;
  dataTime = 1200; …
```


grib_dump without options dumps the coded keys.

grib_dump – examples

```
> grib_dump -O file.grib1
***** FILE: file.grib1
#==============   MESSAGE 1 ( length=3280398 )             ==============
1-4       identifier = GRIB
5-7       totalLength = 3280398
8         editionNumber = 1
=================   SECTION_1 ( length=52, padding=0    ======================
1-3       section1Length = 52
4         table2Version = 128
5         centre = 98 [European Centre for Medium-Range Weather Forecasts (grib1/0.table) ]
6         generatingProcessIdentifier = 145
7         gridDefinition = 255
8         section1Flags = 128 [10000000]
9         indicatorOfParameter = 129 [Geopotential  (m**2 s**-2)  (grib1/2.98.128.table) ]
10        indicatorOfTypeOfLevel = 100 [Isobaric level pressure in hectoPascals(hPa)
                                                        (grib1/local/ecmf/3.table , grib1/3.table) ]
11-12     level = 1000
13        yearOfCentury = 16
14        month = 2
15        day = 27
16        hour = 12
17        minute = 0
18        unitOfTimeRange = 1 [Hour (grib1/4.table) ]
19        P1 = 0 …
```


-O Octet mode. WMO documentation style dump.

grib_dump – examples

```
> grib_dump -OtaH  file.grib1
***** FILE: file.grib1
#==============   MESSAGE 1 ( length=3280398 )             ==============
1-4       ascii identifier = GRIB ( 0x47 0x52 0x49 0x42 )
5-7       g1_message_length totalLength = 3280398 ( 0x32 0x0E 0x0E )
8         unsigned editionNumber = 1 ( 0x01 ) [ls.edition]
======================   SECTION_1 ( length=52, padding=0 )    ======================
1-3       section_length section1Length = 52 ( 0x00 0x00 0x34 )
4         unsigned table2Version = 128 ( 0x80 ) [gribTablesVersionNo]
5         codetable centre = 98 ( 0x62 ) [European Center for Medium-Range Weather
             Forecasts (grib1/0.table) ] [identificationOfOriginatingGeneratingCentre, originatingCentre,
                                                                                ls.centre, centreForTable2]
6         unsigned generatingProcessIdentifier = 139 ( 0x8B ) [generatingProcessIdentificationNumber, process]
7         unsigned gridDefinition = 255 ( 0xFF )
8         codeflag section1Flags = 128 [10000000] ( 0x80 )
9         codetable indicatorOfParameter = 129 ( 0x81 ) [Geopotential  (m**2 s**-2) (grib1/2.98.128.table) ]
10        codetable indicatorOfTypeOfLevel = 100 ( 0x64 ) [Isobaric level pressure in hectoPascals  (hPa)
                                                                  (grib1/3.table) ] [levelType, mars.levtype]
11-12     unsigned level = 1000 ( 0x03 0xE8 ) [vertical.topLevel, vertical.bottomLevel, ls.level, lev,
                                                                                mars.levelist]
13        unsigned yearOfCentury = 11 ( 0x0B )
14        unsigned month = 2 ( 0x02 )
15        unsigned day = 23 ( 0x17 )
16        unsigned hour = 12 ( 0x0C )
17        unsigned minute = 0 ( 0x00 ). . .
```


-O	Octet mode. WMO documentation style dump.
-t	Print type information.
-a	Dump aliases.
-H	Print octet content in hexadecimal format.

grib_dump – examples

```
> grib_dump -D  file.grib1
***** FILE: file.grib1
#==============   MESSAGE 1 ( length=9358 )
  ⋮
======> section GRIB (9358,9358,0)
   0-0 constant ieeeFloats = 0
   ======> section section_0 (0,0,0)
      ----> label empty
   <===== section section_0
   0-4 ascii identifier = GRIB
   4-7 g1_message_length totalLength = 9358
   7-8 unsigned editionNumber = 1 [ls.edition]
   ======> section section_1 (52,52,0)
      ⋮
      36-36 g1date dataDate = 20110223 [mars.date, time.dataDate]
      36-36 evaluate year = 2011
      36-36 time dataTime = 1200 [mars.time]
      36-36 julian_day julianDay = 2.45562e+06
      36-36 codetable stepUnits = 1 [Hour (stepUnits.table) ]
      36-36 concept stepType = instant
      36-36 g1step_range stepRange = 0 [time.stepRange]
      36-36 long_vector startStep = 0
      36-36 long_vector endStep = 0 [stepInHours, mars.step]
      36-36 mars_param marsParam = 129.128 [mars.param]
      36-36 validity_date validityDate = 20110223
      36-36 validity_time validityTime = 1200
    ...
```

In debug mode computed keys are shown

ls.<key> , mars.<key> and time.<key> denote keys in namespaces

-D Debug mode

Practical

- Work in your $SCRATCH

```
cd $SCRATCH
```

- Make a copy of the practicals directory in your $SCRATCH

```
tar -xvf /home/ectrain/trx/ecCodes/grib_tools.tar
```

- This will create a directory in your $SCRATCH containing the GRIB data files for all the practicals
- There is a sub-directory for each practical:

```
ls $SCRATCH/grib_tools
grib_compare  grib_copy   grib_dump   grib_get  grib_ls
grib_set  . . .
```


Login with training ID and do the same…

Practical: using grib_dump

- Use the web documentation to look at the different keys available for type GRIB1 and type GRIB2 messages
  - Identify some keys common to both GRIB1 and GRIB2
- Experiment with using the different grib_dump options (`-O`, `-a` and `-t`). Inspect the GRIB message in the files file1.grib1 and file1.grib2 and identify:
  - the GRIB edition used to encode the messages
  - the (MARS)parameter ID, date, time, forecast step and the grid geometry
- What are the maximum, minimum and average values of the fields?

-a	aliases
-t	type information

grib_ls – list the content of GRIB files

- Use grib_ls to list the content of GRIB files
- Without options grib_ls prints a default list of keys
  - The default list printed is different for GRIB 1 and GRIB 2
- Options exist to specify the set of keys to print or to print keys in addition to the default set
- Output can be ordered
  - e.g. order by ascending or descending step
- grib_ls does not fail if a key is not found
- grib_ls can also be used to find the grid point(s) nearest to a specified latitude-longitude and print the value of the field at that point(s)
  - Modes available to obtain one or four nearest grid points

grib_ls – usage

```
grib_ls [options] grib_file grib file …
```

Options

`-p key[:{s|i|d}],…`	**Keys to print**
`-P key[:{s|i|d}],…`	**Additional keys to print**
`-w key[:{s|i|d}]{=|!=}value,…`	**Where clause**
`-B "key asc, key desc…"`	**Order by: "step asc, centre desc"**
`-n namespace`	**Print all the keys belonging to namespace (ls, parameter, statistics, geography, time, mars, vertical)**
`-m`	**Print MARS keys**
`-W width`	**Minimum column width (default 10)**
`…`	

-p key[:{s/d/l}],key[:{s/d/l}],...
Declaration of keys to print.
For each key a string (key:s) or a double (key:d) or a long (key:l)
type can be requested. Default type is string.
-F format
C style format for floating point values.
-P key[:{s/d/l}],key[:{s/d/l}],...
As -p adding the declared keys at the left of the default list.
-w key[:{s/d/l}]{=/!=}value,key[:{s/d/l}]{=/!=}value,...
Where clause.
Grib messages are processed only if they match all the key/value constraints.
A valid constraint is of type key=value or key!=value.
For each key a string (key:s) or a double (key:d) or a long (key:l)
type can be specified. Default type is string.
-j json output (**works only with –l option**)
-B order by directive
Order by. The output will be ordered according the order by directive.
Order by example: "step asc, centre desc" (step ascending and centre discending)
-l Latitude,Longitude[,MODE,file]
Value close to the point of a Latitude/Longitude.
Allowed values for MODE are:
4 (4 values in the nearest points are printed) Default
1 (the value at the nearest point is printed)
file (file is used as mask. The closer point with mask value>=0.5 is printed)
-s key[:{s/d/l}]=value,key[:{s/d/l}]=value,...
Key/values to set.
For each key a string (key:s) or a double (key:d) or a long (key:l)
type can be defined. By default the native type is set.
e.g. grib_ls -s stepUnits=m msl.grib1
to set stepUnits to minutes so that stepRange values are listed in minutes (rather than hours)
-i index
Data value corresponding to the given index is printed.
-n namespace
All the keys belonging to namespace are printed. Namespaces:
mars
statistics
parameter
Time
Geography
Vertical

-m Mars keys are printed. Same as –n mars
-V Version.
-W width
Minimum width of each column in output. Default is 10.
-M Multi-grib support off. Turn off support for multiple fields in single grib message.
-g Copy GTS header.
-7 Does not fail when the message has wrong length
-x Fast parsing option, only headers are loaded.

grib_ls – examples

Use `-p` option to specify a list of keys to be printed:

```
> grib_ls file.grib2
file.grib2
edition centre date     … gridType    …  typeOfLevel    level    shortName  packingType
2       ecmf   20110226 … reduced_gg …  isobaricInhPa  1000     q          grid_simple
2       ecmf   20110226 … reduced_gg …  isobaricInhPa  850      q          grid_simple
2       ecmf   20110226 … reduced_gg …  isobaricInhPa  700      q          grid_simple
2       ecmf   20110226 … reduced_gg …  isobaricInhPa  500      q          grid_simple
4 of 4 grib messages in file1.grib2

4 of 4 total grib messages in 1 files
```

```
> grib_ls -p centre:l,dataDate,shortName,paramId,typeOfLevel,level file.grib2
file.grib2
Centre      dataDate      shortName     paramId       typeOfLevel     level
98          20110226      q             133           isobaricInhPa   1000
98          20110226      q             133           isobaricInhPa   850
98          20110226      q             133           isobaricInhPa   700
98          20110226      q             133           isobaricInhPa   500
4 of 4 grib messages in file.grib1

4 of 4 total grib messages in 1 files
```


grib_dump/grib_ls with -s

grib_set -s unitOfTimeRange=13,P2=1,timeRangeIndicator=4,yearOfCentury=13,month=5,day=8,hour=0 GRIB1.tmpl temp

grib_ls temp // error

grib_dump temp // error

grib_ls -s stepUnits=m temp // ok. in minutes

grib_ls -s stepUnits=0 -p stepUnits:s temp // check what 0 meant

grib_dump -s stepUnits=m temp // ok

grib_ls – examples

- **When a key is not present in the GRIB file, it returns "not found" for this key**

```
> grib_ls -p my_key  file.grib1

file.grib1
my_key
not found

> echo $?

0
```

exit code returned = 0

- **Similar behaviour to grib_get (see later)**
  - **grib_ls is better for interactive use**
  - **use grib_get within scripts**

Using the ‘where’ option

- **The ‘where option’ `-w` can be used with all GRIB Tools**
- **Constraints are of the form key=value or key!=value**

```
-w key[:{s|i|d}]=value, key[:{s|i|d}]!=value
```

- **Messages are processed only if they match ALL key/value constraints**
- **Values separated by / represent “OR” condition**

```
> grib_ls -w levelType=pl file.grib1
...
> grib_ls -w step!=6,level=700/850 file.grib1
...
> grib_ls -w count=3 file.grib1
```


Key type specification doesn’t seem any longer necessary.

I believe in recent versions the tools have been made smart so that you don't have to specify the type identifier anymore and I can't think of an example right now when that would be necessary. It is however useful to control the output values of a key, i.e. if you want to display the string representation of centre or the integer one.

Practical: using grib_ls

- Use grib_ls to inspect the files msl.grib1 and msl.grib2
  - Which keys does grib_ls show by default for the two files ?
  - What fields do they contain ?
- Use grib_ls to print the MARS keys
- Use grib_ls with other namespaces
- Use grib_ls to order the output by descending step
- Use grib_ls to print the centre, dataDate, stepRange, levelType, shortName and paramId for both files
  - Experiment with both –P and –p options and 'key:i', 'key:s'

Finding nearest grid points with grib_ls

- **The value(s) of a GRIB field close to the point of a Latitude/Longitude can be found with grib_ls**

```
grib_ls -l Latitude,Longitude[,MODE,file] grib_file
```

`MODE` **Can take the values**

4	**Print values at the 4 nearest grid points (default)**
1	**Print value at the closest grid point**

`file` **Specifies a GRIB file to use as a mask**
The closest *land* point (with mask ≥ 0.5) is printed

- **GRIB files specified must contain grid point data**

Practical: using grib_ls –l

- **The file msl.grib1 contains the mean sea-level pressure from the EPS control forecast at 6-hourly time steps for the first 24 hours on a N100 regular Gaussian grid**
- **Find the value of the MSLP at the grid point nearest to ECMWF (Lat 51.42°N, Lon 0.95° W) at each forecast step**
  - **What is the lat-lon value of the grid point nearest to ECMWF ?**
  - **How far is the chosen grid point from ECMWF ?**
- **Change the command used to output only the forecast step and the MSLP value at the nearest grid point**
- **Change the command to output the MSLP values at the four grid points nearest to ECMWF**
- **Use the file lsm.grib1 to provide a land-sea mask**
  - **Are all four nearest grid points land points (mask ≥ 0.5) ?**

grib_get – get key / value pairs

- Use grib_get to get the values of one or more keys from one or more GRIB files – very similar to grib_ls
- By default grib_get fails if an error occurs (e.g. key not found) returning a non-zero exit code
  - Suitable for use in scripts to obtain key values from GRIB messages
  - Can force grib_get not to fail on error
- Options available to get all MARS keys or all keys for a particular namespace
  - Can get other keys in addition to the default set
- Format of floating point values can be controlled with a C-style format statement

grib_get – usage

```
grib_get [options] grib_file grib_file ...
```

- Options

`-p key[:{s|i|d}],…`	Keys to get
`-P key[:{s|i|d}],…`	Additional keys to get with `-m`, `-n`
`-w key[:{s|i|d}]{=/!=}value,` …	Where option
`-s key[:{s|i|d}]=value,…`	Keys to set
`-m`	Get all MARS keys
`-n namespace`	Get all keys for `namespace`
`-l lat,lon[,MODE,FILE]`	Value(s) nearest to lat-lon point
`-F format`	Format for floating point values
`-f`	Do *not* fail on error

Shahram will explain namespace next mornig

grib_get – examples

- To get the centre of the first (**count=1**) GRIB message in a file (both as a 'string' and an 'integer')

```
> grib_get -w count=1 -p centre f1.grib1
ecmf

> grib_get -w count=1 -p centre:i f1.grib1
98
```

- grib_get fails if there is an error

```
> grib_get -p mykey f1.grib1
ECCODES ERROR   :  Key/value not found

> echo $?
246
```

returns the exit code from the previous command

grib_get – examples

- To get all the MARS keys, optionally printing the shortName

```
> grib_get -m f1.grib1
g sfc 20150223 1200 0 167.128 od an oper 0001

> grib_get -m -P shortName f1.grib1
2t g sfc 20150223 1200 0 167.128 od an oper 0001
```

- To get all keys belonging to the statistics namespace

```
> grib_get -n statistics f1.grib1
314.24 214.613 277.111 21.0494 41379.8 2.48314e-05 0
```

- grib_get –m is the same as grib_get –n mars

grib_get – controlling output format

- The format of floating point values can be controlled by using a C-style format statement with the **-F** option
  - **-F "%.4f"** - Decimal format with 4 decimal places (1.2345)
  - **-F "%.4e"** - Exponent format with 4 decimal places (1.2345E-03)

```
> grib_get -F "%.6f" -p maximum f1.grib1

314.240280

> grib_get -F "%.4e" -p maximum f1.grib1

3.1424e+02
```

- Default format is **-F "%.10e"**

grib_get – stepRange and stepUnits

- The step is always printed as an integer value
- By default the units of the step are printed in hours
- To obtain the step in other units set the stepUnits appropriately with the **-s** option

```
> grib_get -p stepRange f1.grib1
6
12

> grib_get -s stepUnits=m -p stepRange f1.grib1
360
720
```

stepUnits can be s, m, h, 3h, 6h, 12h, D, M, Y, 10Y, 30Y, C

Finding nearest grid points with grib_get

- The value of a GRIB field close to a specified point of Latitude/Longitude can be found with grib_get
  - Works in the same way as grib_ls

```
> grib_get -l 52.0,-1.43 f1.grib1

273.58 272.375 273.17 273.531


> grib_get -F "%.5f" -P stepRange -l 52.0,-1.43,1 f1.grib1

0 272.37505
```

- GRIB files specified must contain grid point data

Getting data values at a grid point

- The value of a GRIB field at a particular grid point can be printed using grib_get with the **-i** option
- For example, find the index of a nearest grid point with grib_ls and then use this with grib_get to build a list of values at that point:

```
> grib_get -F "%.2f" -i 2159 -p step,dummy:s f1.grib1

6 99429.31
12 99360.25
18 99232.31
24 99325.56
```

Forces a space between step and value

- Also returns a value for non-grid point data !

grib_get_data – print data values

- Use grib_get_data to print a list of latitude, longitude (for grid point data) and data values from one or more GRIB files
- The format of the output can be controlled by using a C-style format statement with the **-F** option
  - **-F "%.4f"** - Decimal format with 4 decimal places (1.2345)
  - **-F "%.4e"** - Exponent format with 4 decimal places (1.2345E-03)
  - The default format is **-F "%.10e"**
- By default missing values are not printed
  - A user-provided string can be printed in place of any missing values
- By default grib_get_data fails if there is an error
  - Use the **-f** option to force grib_get_data not to fail on error

grib_get_data – usage

```
grib_get_data [options] grib_file grib_file ...
```

- Options

`-p key[:{s|i|d}],…`	Keys to print
`-w key[:{s|i|d}]{=/!=}value,…`	Where option
`-m missingValue`	Specify missing value string
`-F format`	C-style format for output values
`-f`	Do *not* fail on error
`...`	

grib_get_data – example

```
> grib_get_data -F "%.4f" f1.grib1
Latitude, Longitude, Value
   81.000     0.000 22.5957
   81.000     1.500 22.9009
   81.000     3.000 22.8359
   81.000     4.500 22.3379
   81.000     6.000 21.5547
   81.000     7.500 20.7344
   81.000     9.000 19.8916
   81.000    10.500 18.5747
   81.000    12.000 17.2578
   81.000    13.500 16.1343
   81.000    15.000 14.9785
   81.000    16.500 13.8296
...
```

Format option applies to values only - not to the Latitudes and Longitudes

grib_get_data – missing values example

```
> grib_get_data -m XXXXX -F "%.4f" f1.grib1
Latitude, Longitude, Value
...
   81.000    90.000 9.4189
   81.000    91.500 8.6782
   81.000    93.000 XXXXX
   81.000    94.500 XXXXX
   81.000    96.000 XXXXX
   81.000    97.500 XXXXX
   81.000    99.000 6.7627
   81.000   100.500 7.4097
   81.000   102.000 7.9307
...
```

Missing values are printed as XXXXX

Practical: using grib_get & grib_get_data

1. Use grib_get to obtain a list of all the pressure levels available for parameter T in the file tz_an_pl.grib1
2. Use grib_get to print the shortName, dataTime, dataDate and level for the 500 &1000 hPa levels only
3. Use grib_get to print the stepRange for the fields in the file surface.grib1 in (a) hours (b) minutes and (c) seconds
4. Use grib_get_data to print the latitude, longitude and values for the first field in surface.grib1
   - Output results in decimal format with 5 decimal places
   - Output results in exponential format with 10 decimal places
5. Use grib_get_data to print the data values for the temperature at 500 hPa only from the file tz_an_pl.grib1
   - Make sure you print only the data for T500 ! What is printed ?