GRIB tools examples

grib_compare examples

1. The default behaviour for grib_compare without any option is to perform a bit by bit comparison of the two messages. If the messages are found to be bitwise different then grib_compare switches to a "key based" mode to find out which coded keys are different. To see how grib_compare works we first set the shortName=2d (2 metre dew point temperature) in the file regular_latlon_surface.grib1

```
> grib_set -s shortName=2d regular_latlon_surface.grib1 2d.grib1
```

grib_copy examples

1. To copy only the pressure levels from a file

```
>  grib_copy -w levtype=pl ../data/tigge_pf_ecmwf.grib2 out.grib
```

2. To copy only the fields that are not on pressure levels from a file

```
>  grib_copy -w levtype!=pl ../data/tigge_pf_ecmwf.grib2 out.grib
```

3. To copy only the first three fields from a file

```
>  grib_copy -w count=1/2/3 ../data/tigge_pf_ecmwf.grib2 out.grib
```

4. A grib_file with multi field messages can be converted in single field messages with a simple grib_copy.

```
>  grib_copy multi.grib simple.grib
```

5. Use the square brackets to insert the value of a key in the name of the output file (This is a good way to split a large GRIB file)

```
>  grib_copy in.grib 'out_[shortName].grib'
```

   Note: we need to quote the name of the output so the shell does not interpret the square brackets

6. To copy fields whose typeOfLevel is either 'surface' or 'meanSea'

```
>  grib_copy -w typeOfLevel=surface/meanSea orig.grib out.grib
```

7. To copy selected fields and apply sorting (sorted by level in ascending order)

```
>  grib_copy -w typeOfLevel=heightAboveGround -B'level:i asc' tigge_af_ecmwf.grib2 out.grib
```

   Note: we need to specify the ':i' to get a numerical sort. By default values are sorted as strings so a level of 100 would come before 20!

grib_dump examples

1. To dump in a WMO documentation style with hexadecimal octet values (-H).

```
> grib_dump -OH ../data/reduced_gaussian_model_level.grib1
```

2. To add key aliases and type information.

```
> grib_dump -OtaH ../data/reduced_gaussian_model_level.grib1
```

3. To obtain all the key names (computed keys included) available in a grib file.

```
>  grib_dump -D ../data/regular_latlon_surface.grib1
```

grib_filter examples

1. The grib_filter processes sequentially all grib messages contained in the input files and applies the rules to each one of them. Input messages can be written to the output by using the "write" statement. The write statement can be parameterised so that output is sent to multiple files depending on key values used in the output file name. If we write a rules_file containing the only statement:

```
write "../data/split/[centre]_[date]_[dataType]_[levelType].grib[editionNumber]";
```

   Applying this rules_file to the "../data/tigge_pf_ecmwf.grib2" grib file we obtain several files in the ../data/split directory containing fields split according to their key values

```
> grib_filter rules_file ../data/tigge_pf_ecmwf.grib2
> ls ../data/split
ecmf_20060619_pf_sfc.grib2
ecmf_20060630_pf_sfc.grib2
ecmf_20070122_pf_pl.grib2
ecmf_20070122_pf_pt.grib2
ecmf_20070122_pf_pv.grib2
ecmf_20070122_pf_sfc.grib2
```

2. The key values in the file name can also be obtained in a different format by indicating explicitly the type required after a colon.

   - :i for integer
   - :d for double
   - :s for string

   The following statement works in a slightly different way from the previous example, including in the output file name the integer values for centre and dataType.

```
write "../data/split/[centre:i]_[date]_[dataType:i]_[levelType].grib[editionNumber]";
```

   Running the same command again we obtain a different list of files.

```
> grib_filter rules_file ../data/tigge_pf_ecmwf.grib2
> ls ../data/split
98_20060619_4_sfc.grib2
98_20060630_4_sfc.grib2
98_20070122_4_pl.grib2
98_20070122_4_pt.grib2
98_20070122_4_pv.grib2
98_20070122_4_sfc.grib2
```

3. Other statements are allowed in the grib_filter syntax:

    - if (condition) { block of rules } else { block of rules } The condition can be made using ==,!= and joining single block conditions with || and && The statement can be any valid statement also another nested condition
    - set keyname = keyvalue;
    - print "string to print also with key values like in the file name"
    - transient keyname1 = keyname2;
    - comments beginning with #
    - defined(keyname) to check if a key is defined in a message
    - missing(keyname) to check if the value of the key is set to MISSING (Note: This does not apply to codetable keys)
    - To set a key value to MISSING, use 'set key=MISSING;' (note the case)
    - You can also make an assertion with 'assert(condition)'. If condition is false, it will abort the filter.

    A complex example of grib_filter rules is the following to change temperature in a grib edition 1 file.

```
# Temperature
if ( level == 850 && indicatorOfParameter == 11 ) {
    print "found indicatorOfParameter=[indicatorOfParameter] level=[level] date=[date]";
    transient oldtype = type ;
    set identificationOfOriginatingGeneratingSubCentre=98;
    set gribTablesVersionNo = 128;
    set indicatorOfParameter = 130;
    set localDefinitionNumber=1;
    set marsClass="od";
    set marsStream="kwbc";
    # Negatively/Positively Perturbed Forecast
    if ( oldtype == 2 || oldtype == 3 ) {
      set marsType="pf";
      set experimentVersionNumber="4001";
    }
    # Control Forecast
    if ( oldtype == 1 ) {
      set marsType="cf";
      set experimentVersionNumber="0001";
    }
    set numberOfForecastsInEnsemble=11;
    write;
    print "indicatorOfParameter=[indicatorOfParameter] level=[level] date=[date]";
    print;
}
```

4. Here is an example of an IF statement comparing a key with a string. Note you have to use the "is" keyword for strings and not "==", and to negate you add the "!" before the whole condition:

```
# Select Geopotential Height messages which are not on a Reduced Gaussian Grid
if (shortName is "gh" && !(gridType is "reduced_gg" )) {
    set step = 72;
}
```

5. The switch statement is an enhanced version of the if statement. Its syntax is the following:

```
switch (key1) {
    case val1:
        # statements
    case val2:
        # statements
    default:
        # statements
}
```

The value of the key given as argument to the switch statement is matched against the values specified in the case statements. If there is a match, then the statements corresponding to the matching case are executed. Otherwise, the default case is executed. The default case is mandatory if the case statements do not cover all the possibilities. The "~" operator can be used to match "anything". The following is an example of the switch statement:

```
print 'Processing paramId=[paramId] [shortName] [stepType]';
switch (shortName) {
    case 'tp' :
        set stepType='accum';
        print 'Message #[count]: Total Precipitation';
    case '10u' :
        set typeOfLevel='surface';
        print 'Message #[count]: 10m U-Wind';
    default:
}
```

grib_get examples

1. grib_get fails if a key is not found.

```
> grib_get -p nosuchkey ../data/tigge_pf_ecmwf.grib2
ECCODES ERROR   :  Key/value not found
```

grib_get_data examples

1. If you want to define your missing value=1111 and to print the string 'missing' in place of it

```
> grib_get_data -m 1111:missing ../data/reduced_gaussian_model_level.grib2
```

2. If you want to print the value of other keys with the data value list

```
> grib_get_data -p centre,level,step ../data/reduced_gaussian_model_level.grib2
```

grib_index_build examples

1. By default grib_index_build will index on the MARS keys.

```
>  grib_index_build ../data/reduced*.grib1 ../data/regular*.grib1 ../data/reduced*.grib2
```

2. To specify a custom list of keys to index on, use the -k option.

```
>  grib_index_build -k paramId,dataDate ../data/reduced*.grib1 ../data/regular*.grib1 ../data/reduced*.
grib2
```

grib_ls examples

1. Without options a default list of keys is printed. The default list is different depending on the type of grib message.

```
>  grib_ls ../data/reduced*.grib1 ../data/regular*.grib1 ../data/reduced*.grib2
```

2. To print offset and count number in file use the keys offset and count Also the total count in a set of files is available as countTotal

```
>  grib_ls -p offset,count,countTotal ../data/reduced*.grib1
```

3. To list only a subset of messages use the -w (where option). Only the pressure levels are listed with the following line.

```
>  grib_ls -w levelType=pl ../tigge_pf_ecmwf.grib2
```

4. All the grib messages not on pressure levels are listed as follows:

```
>  grib_ls -w levelType!=pl ../tigge_pf_ecmwf.grib2
```

5. To get the closest grid point to a latitude/longitude.

```
> grib_ls -l 51.46,-1.33,1 -p paramId,name ../data/reduced_gaussian_surface.grib2
../data/reduced_gaussian_surface.grib2
paramId     shortName    value
167         2t          282.002
1 of 1 messages in ../data/reduced_gaussian_surface.grib2

1 of 1 total messages in 1 files
Input Point: latitude=51.46  longitude=-1.33
Grid Point chosen #3 index=749 latitude=51.63 longitude=0.00 distance=93.81 (Km)
Other grid Points
- 1 - index=845 latitude=48.84 longitude=0.00 distance=306.86 (Km)
- 2 - index=944 latitude=48.84 longitude=356.40 distance=333.66 (Km)
- 3 - index=749 latitude=51.63 longitude=0.00 distance=93.81 (Km)
- 4 - index=844 latitude=51.63 longitude=356.25 distance=168.37 (Km)
```

6. To get a list ordered by the 'level' key (ascending order).

```
>  grib_ls -B 'level:i asc' tigge_af_ecmwf.grib2
```

Note: we need to specify the ':i' to get a numerical sort. By default values are sorted as strings so a level of 100 would come before 20!

grib_set examples

1. To set productDefinitionTemplateNumber=2 only for the fields with productDefinitionTemplateNumber=11

```
> grib_set -s productDefinitionTemplateNumber=2 -w productDefinitionTemplateNumber=11 ../data
/tigge_pf_ecmwf.grib2 out.grib2
```

2. To set productDefinitionTemplateNumber=2 only for the fields for which productDefinitionTemplateNumber is not equal to 11

```
> grib_set -s productDefinitionTemplateNumber=2 -w productDefinitionTemplateNumber!=11 tigge_pf_ecmwf.
grib2 out.grib2
```

3. When a key is not used all the bits of its value should be set to 1 to indicate that it is missing. Since the length (number of octet) is different from a key to another, the value that we have to code for missing keys is not unique. To give an easy way to set a key to missing a string "missing" or "MISSING" is accepted by grib_set as follows:

```
> grib_set -s scaleFactorOfFirstFixedSurface=missing,scaledValueOfFirstFixedSurface=MISSING ../data
/regular_latlon_surface.grib2 out.grib2
```

   Since some values can not be set to missing you can get an error for those keys.

4. To set scaleFactorOfSecondFixedSurface to missing only for the fields for which scaleFactorOfSecondFixedSurface is not missing:

```
> grib_set -s scaleFactorOfSecondFixedSurface=missing -w scaleFactorOfSecondFixedSurface!=missing
tigge_pf_ecmwf.grib2 out.grib2
```

5. It is possible to produce a GRIB edition 2 file from a GRIB edition 1 by just changing the edition number with grib_set. However it is important that you carefully inspect the output and check the information is correctly translated.

```
grib_set -s edition=2 ../data/reduced_gaussian_pressure_level.grib1 out.grib2
```

   For more details please see: Converting edition 1 to 2 - ecCodes GRIB FAQ

6. With grib edition 2 is possible to compress data using the jpeg algorithm. To change packing algorithm from grid_simple (simple packing) to grid_jpeg (jpeg2000 packing):

```
> grib_set -s packingType=grid_jpeg ../data/regular_gaussian_model_level.grib2 out.grib2
```

7. It's possible to ask ecCodes to calculate the number of bits per value needed to pack a given field with a fixed number of decimal digits of precision. For example if we want to pack a temperature expressed in Kelvin with 1 digits of precision after the decimal point we can set changeDecimalPrecision=1

```
> grib_set -s changeDecimalPrecision=1 ../data/regular_latlon_surface.grib2 ../data/out.grib2
```

grib_to_netcdf examples

1. Produce a NetCDF file from grib edition 1

```
> grib_to_netcdf -o output.nc input.grib1
```

2. If your grib file has analysis and 6-hour forecast, then ignore keys 'type' and 'step'. Thus type=an/fc and step=00/06 will not be considered as netcdf dimensions.

```
> grib_to_netcdf -I type,step -o output.nc input.grib
```

3. Do not use time of validity. If time of validity is used, it means the 1D time coordinate is considered as date+time+step, otherwise 3 different dimensions are created. The default behaviour is to use the time of validity.

```
> grib_to_netcdf -T -o output.nc input.grib
```

4. Produce NetCDF with data type of FLOAT (32bit floating-point, for higher precision). Note these types were chosen to provide a reasonably wide range of trade-offs between data precision and number of bits required for each value

```
> grib_to_netcdf -D NC_FLOAT -o output.nc input.grib
```

5. Set the netcdf dimension 'time' to be unlimited i.e. time can have unlimited length so variables using this dimension can grow along this dimension.

```
> grib_to_netcdf -u time -o output.nc input.grib
```