GRIB API – advanced tools

Computer User Training Course 2014

Paul Dando

User Support Section

advisory@ecmwf.int

Overview

- grib_filter
  - Introduction
  - Rules syntax
  - Examples
  - Practical
- grib_to_netcdf
  - Usage
  - Examples
  - Practical

grib_filter – introduction

- GRIB API advanced command-line tool
- Iterates over all the messages in the input
- Applies a set of user defined rules to each message
- The rules are formed using a macro language GRIB API provides
- Note that the macro language does not have the capabilities of a full-blown programming language

grib_filter – introduction

- Access data inside a message through keys
- Print contents of a message
- Set values inside a message
- Use control structures (`if`, `switch`)
- Write a message to disk

grib_filter – usage

```
grib_filter [-o out_file] rules_file in_file1 in_file2 …
```

- Each field from the input files is processed and the rules contained in the rules_file are applied to it
- A GRIB message is written to an output file only if a write instruction is applied to it
- Each instruction in the rules_file must end with a semicolon "`;`"
- Syntax errors in the rules_file are reported with their line number

Rules syntax – print statement

- `print “some text”; # this is a comment`
- `print “some text [key]”;`
  - Print to the standard output.
  - Retrieve the value of the keys in squared brackets.
  - If a key is not found in the message then the value of `[key]` will be displayed as “undef”.
  - `[key]` -> native type
  - `[key:l]` -> integer (the “el” is for “long”!)
  - `[key:s]` -> string
  - `[key:d]` -> double
  - `[key!c%F’S’]` -> arrays: c->columns F->format (C style) S->separator
- `print (“filename”) “some text [key]”;`

Example 1 – using print

```
# A simple print

print "ed = [edition] centre is [centre:s] = [centre:l]";
```

```
> grib_filter rule.filter x.grib1

ed = 1 centre is ecmf = 98
```

Example 2 – formatted print

```
# one column 3 decimal digits
print "[distinctLatitudes!1%.3f]";
```

```
> grib_filter rule.filter x.grib1
-90.000
-88.500
-87.000
-85.500
…
```

Example 3 – print with separator

```
# three columns 5 decimal digits comma separated
print "[latLonValues!3%.5f',']";
```

```
> grib_filter rule.filter x.grib1
90.00000,0.00000,1.00000,
90.00000,1.50000,1.00000,
90.00000,3.00000,1.00000,
…
```

Rules syntax – write statement

- `write;`
  - Writes the current message to the output file defined in the command line with the option `–o` (grib_filter –o outfile rules_file grib_file)
  - If the `–o` option is not specified, the default value “filter.out” is used

- `write “filename_[key]”;`
  - Writes the current message to the file “`filename_[key]`” where the key in square brackets is replaced with its value retrieved from the message
  - Important “side effect” is that if two messages have different values for key they are also written to different files

Example 4 – write statement

```
# Creating multiple files
write “[centre]_[dataDate]_[step].grib[edition]”;
```

```
> grib_filter rule.filter x.grib1
> ls
ecmf_20080213_0.grib1
ecmf_20080213_6.grib1
ecmf_20080213_12.grib1
ecmf_20080213_24.grib1
```

Rules syntax – append statement

- `append;`
  - Appends the current message to the output file defined in the command line with the option **–o** (grib_filter –o outfile rules_file grib_file).
  - If the **–o** option is not specified, the default value "filter.out" is used

- `append "filename_[key]";`
  - Appends the current message to the file "**filename_*[key]***" where the key in square brackets is replaced with its value retrieved form the message
  - If the file is created if it does not exist
  - If two messages have different values for key, they are appended to different files

Example 5 – append statement

```
append;
```

```
> grib_count out.grib
> 1
>
> grib_filter rule.filter -o out.grib in.grib
>
> grib_count out.grib
> 2
```

Rules syntax – setting keys

- `set key1 = key2 ;        # set key1 to the value of key2`
- `set key = {val1,val2,val3,val4} ; # set an array key`
- `set key = "string" ;             # set key to a string`
- `set key = expression ;       # set key to an expression`
- `set key = MISSING ;       # set value of key to missing`
- expression operators :

`==`	equal to
`!=`	not equal to
`is`	equals to for strings
`||`	or
`&&`	and
`!`	not
`* / + -`	arithmetic operators
`( )`	

Example 6 – setting a key

```
set edition = 2;

write “[file][edition]”;
```

```
> grib_filter rule.filter x.grib

> ls

x.grib

x.grib2
```

Example 7 – setting an array key

```
set values = {12.2,14.8,13.7,72.3};
print “values = { [values] }”;
write “[file].[edition]”;
```

```
> grib_filter rule.filter x.grib
values = {  12.2 14.8 13.7 72.4 }
```

Rules syntax – transient keys

- `transient key1 = key2;`
  - Defines the new key1 and assigns to it the value of key2
- `transient key1 = "string";`
- `transient key1 = expression ;`
- expression operators:

`==`	equal to
`!=`	not equal to
`is`	equals to for strings
`||`	or
`&&`	and
`!`	not
`* / + -`	arithmetic operators
`( )`	

Example 8 – transient keys

```
transient mystep = step + 24;

print “step = [step] mystep = [mystep]”;
```

```
> grib_filter rule.filter x.grib

step = 24 mystep = 48
```

Practicals

- To get the material for these practicals:

```
cd $SCRATCH/grib_tools/grib_filter
```

- Run the filter files ‘print.filter’, ‘write.filter’, ‘transient.filter’ on ‘tigge.grib’.
- Comment/uncomment the instructions one by one to see the different behaviours.

Rules syntax – if statement

- `if ( expression ) { instructions }`
- `if ( expression ) { instructions }`
  `else { instructions }`

- **Expression operators:**

`==`	equal to
`!=`	not equal to
`is`	equals to for strings
`||`	or
`&&`	and
`!`	not
`* / + -`	arithmetic operators
`( )`	

Example 9 – if statement

```
if (localDefinitionNumber == 1) {
   set edition = 2;
   write;
}
```

```
> grib_filter -o out.grib2 rule.filter x.grib1
> ls
out.grib2
```

Rules syntax – switch statement

- Alternate version of an ‘if-else’ statement.
- More convenient to use when you have code that needs to choose a path from many to follow.

```
switch (var) {
      case val1:
            # set of actions
            ...
      case val2:

            # set of actions
            ...
      default:
            # default block of actions

}
```

Example 10 – switch statement

```
print "processing [paramId] [shortName] [stepType]";
switch (shortName) {
    case "tp" :
          set stepType="accum";
    case  "sp" :
          set typeOfLevel="surface";
    default:
          print "Unexpected parameter";
}
write;
```

Example 11

```
if (centre is "lfpw" &&
   (indicatorOfParameter == 6 ||
    indicatorOfParameter == 11 ||
    indicatorOfParameter == 8) )
{
   if (step!=0) {
      set typeOfGeneratingProcess=0;
      set typeOfProcessedData=0;
   } else {
     # Other steps
     set typeOfProcessedData=1;
 …
```

```
…
  switch (typeOfLevel) {
    case "hybrid":
      set changeDecimalPrecision=1;
    case "surface":
      set changeDecimalPrecision=2;
    case "isobaricInhPa":
      if (level > 300) {
         print "level > 300";
         set level = level*2 + 15;
      } # end if (level > 300)
    default:
      print "Unknown level type!";
    } # end switch (typeOfLevel)
 } # end if (step!=0)
 write;
} # end main if
```

grib_to_netcdf – convert to netCDF

- Use grib_to_netcdf to convert GRIB messages to netCDF
- Input GRIB fields must be on a regular grid
  - gridType=regular_ll or gridType=regular_gg
- Options allow user to specify the netCDF data type:
  - NCBYTE, NC_SHORT, NC_INT, NC_FLOAT or NC_DOUBLE
  - NC_SHORT is the default
- Options allow the user to specify the reference date
  - Default is 19000101
- Used in the MARS web interface and the public Data Servers to provide data in netCDF

grib_to_netcdf – usage

```
grib_to_netcdf [options] grib_file grib_file …
```

- Options

`-o output_file`	Output netCDF file
`-R YYYYMMDD`	Use YYYYMMDD as reference date
`-D NC_DATATYPE`	netCDF data type
`-f`	Do *not* fail on error
`...`	

grib_to_netcdf – examples

- To convert the fields in file.grib1 to netCDF

```
> grib_to_netcdf -o out.nc file.grib1
grib_to_netcdf: Processing input file 'file.grib1'.
grib_to_netcdf: Found 1 GRIB fields in 1 file.
grib_to_netcdf: Creating netcdf file 'out.nc'
grib_to_netcdf: NetCDF library version: "3.6.3" of
  Aug 31 2010 17:29:46 $
grib_to_netcdf: Defining variable 't'.
grib_to_netcdf: Done.

> ls -s out.nc
132 out.nc
```

grib_to_netcdf – examples

- To convert the fields in file.grib1 to netCDF with data type set to NC_FLOAT

```
> grib_to_netcdf -D NC_FLOAT -o out.nc file.grib1
grib_to_netcdf: Processing input file 'file.grib1'.
grib_to_netcdf: Found 1 GRIB field in 1 file.
grib_to_netcdf: Creating netcdf file 'out.nc'
grib_to_netcdf: NetCDF library version: "3.6.3" of
  Aug 31 2010 17:29:46 $
grib_to_netcdf: Defining variable 't'.
grib_to_netcdf: Done.

> ls -s out.nc
260 out.nc
```

Output netCDF file is about twice the size

Practical – grib_to_netcdf

1. Use grib_to_netcdf to convert the GRIB messages in file1.grib to netCDF
   - Try with both the default data type (NC_SHORT) and NC_FLOAT
   - Check the data values in each case with ncdump
2. Repeat but set the reference date to 25 February 2014
   - Check the output with ncdump and compare with the previous exercise
3. Use grib_to_netcdf to convert the GRIB messages in file2.grib to netCDF
   - What happens … and why ?

Questions ?