

ecCodes: using bufr_filter

Introduction

Computer User Training Course 2018

Shahram Najm

Development Section

Forecast Department



bufr_filter – introduction

- ecCodes 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 used by ecCodes
- Note that the macro language does not have the capabilities of a full-blown programming language

bufr_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

bufr_filter – usage

```
bufr_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 BUFR 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
- Always put “-o out_file” before the other options
- You need to `set unpack=1` to decode (unpack) the data section
- You need to `set pack=1` to encode (pack) the data section

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:i]` -> integer
 - `[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 "edition=[edition], centre=[centre:s] (= [centre:i])";
```

```
> bufr_filter rule.filter x.bufr  
edition=4, centre=ecmf (=98)
```

Example 2 – formatted print

```
# 1 column and 4 decimal digits  
set unpack = 1;    # unpack to decode data section  
print "[second!1%.4f]";
```

```
> bufr_filter rule.filter x.bufr
```

```
7.5470
```

```
23.5430
```

```
31.5430
```

```
...
```

Example 3 – print with separator

```
# 5 columns, 4 decimal digits and comma separated  
set unpack = 1; # unpack to decode data section  
print "[latitude!5%.4f', ']";
```

```
> bufr_filter rule.filter x.bufr  
43.1196,43.5967,43.9777,44.2931,44.5612,  
44.7942,45.0002,45.1854,45.3538,45.5090,  
45.6532,45.7886,45.9168,46.0391,46.1565,  
...
```


Rules syntax – write statement

- **write;**

- Writes the current message to the output file defined in the command line with the option `-o`
`bufr_filter -o outfile rules_file bufr_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
- If two messages have different values for *[key]* they are also written to different files

Example 4 – write statement

```
# Creating multiple files  
write "out__[satelliteID]_[typicalYear].bufr[edition]";
```

```
> bufr_filter rule.filter x.bufr
```

```
> ls
```

```
out__248_2012.bufr3
```

```
out__285_2009.bufr3
```

```
...
```

Rules syntax – append statement

- **append;**

- Appends the current message to the output file defined in the command line with the option `-o`
`bufr_filter -o outfile rules_file bufr_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 from the message
- 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;
```

```
> bufr_count out.bufr
```

```
> 1
```

```
>
```

```
> bufr_filter -o out.bufr rule.filter in.bufr
```

```
>
```

```
> bufr_count out.bufr
```

```
> 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
- expression operators :

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

Example 6 – setting a header key

```
set typicalMonth = 3;  
write "[file][edition]";
```

```
> bufr_filter rule.filter x.bufr  
> ls  
x.bufr  
x.bufr4  
> bufr_get -p typicalMonth,typicalDate x.bufr*  
11 20121102  
3 20120302
```

Example 7 – setting an array key

```
set unpack = 1; # Need to unpack the data section
set longitude = {-1.57e+02, -1.56e+02, ...};
print "longitude = { [longitude] }";
set pack = 1;
write "[file].[edition]";
```

```
> bufr_filter rule.filter x.bufr
longitude = { -157 -156 ... }
```

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;`
- `set key1 = key2;` # change an existing transient
- expression operators:

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

Example 8 – transient keys

```
set unpack = 1;  
transient statid = 1000*blockNumber + stationNumber;  
print "statid=[statid] t2=[airTemperatureAt2M]";  
write;
```

```
> bufr_filter rule.filter x.bufr  
  
statid=1001 t2=274.5  
statid=1003 t2=268.4
```

Rules syntax – if statement

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

There is no 'else if' - you have to create a new 'if' block

- Expression operators:

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

Example 9 – if statement

```
if (bufrHeaderCentre == 98) {  
  # This is ECMWF  
  set edition = 4;  
  write;  
}
```

```
> bufr_filter -o out.bufr rule.filter in.bufr  
> bufr_get -p edition in.bufr out.bufr  
3  
4
```

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 (key) {  
    case val1:  
        # set of actions  
        ...  
    case val2:  
        # set of actions  
        ...  
    default:  
        # default block of actions  
}
```

*default: case
is mandatory
even if empty*

Example 10 – switch statement

```
print "processing [file], msg #[count]";
switch (satelliteID) {
    case 207 :
        print "Processing XXX... ";
        ...
    case 209 :
        print "Processing YYY... ";
        ...
    default:
        print "Unexpected satellite ID [satelliteID]";
}
write;
```

Example 11

```
# A rather contrived example! 😊
if (centre is "lfpw" &&
    (year == 2016 ||
     year == 2011))
{
  if (month != 1 && day < 25) {
    set relativeHumidity=27;
  } else {
    # Other values
    set relativeHumidity=28;
  }
}
```

Rules syntax – 'defined' function

- **defined(key)**
- Returns true if the given key exists in the message being processed, false otherwise

```
set unpack=1;
if (defined(airTemperature)) {
    print "File [file], msg #[count] has airTemperature";
    print "[airTemperature->percentConfidence]";
} else {
    print "File [file], msg #[count] does not have airTemperature";
}
```

```
> bufr_filter rule.filter *.bufr
File aaen_55.bufr, message #1 does not have airTemperature
File temp_101.bufr, message #2 has airTemperature
70 70 ...
```

Rules syntax – 'assert' statement

- `assert (condition) ;`
- If the condition evaluates to false then the filter will abort

```
# This filter should be run on BUFR edition 4 only.  
# Abort otherwise  
  
assert (edition == 4);  
  
...
```

```
> bufr_filter -o out.bufr rule.filter x.bufr  
ECCODES ERROR : Assertion failure:
```


Practicals

- To get the material for these practicals:

```
cd $SCRATCH
cp -r ~trx/ecCodes/2018/bufr_tools_filter1 ./
cd bufr_tools_filter1
```

1. Run bufr_filter with the rules files 'print.filter', 'write.filter', 'transient.filter' on 'aaen_55.bufr'.
2. Comment/uncomment the instructions one by one to see the different behaviours.
3. Convert any edition 3 messages in 'aaen_55.bufr' and 'ahws_139.bufr' to edition 4
4. How would you check the conversions worked?