bufr_filter practicals I Solutions

Roberto Ribas

Forecast department

Copernicus Production Section

Preparations:

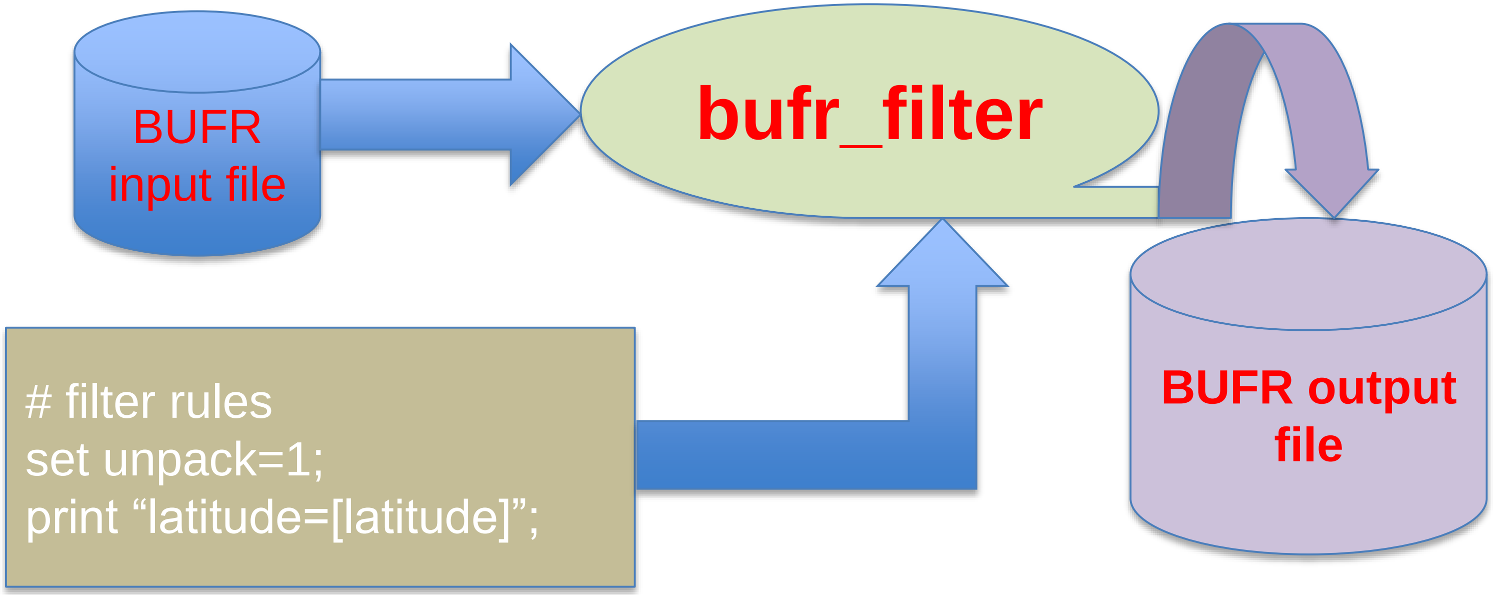
Copy the **bufr_tools_filter_adv** directory to your local directory.

```
cd $SCRATCH
cp –r ~trx/ecCodes/2018/bufr_tools_filter_adv  .
cd bufr_tools_filter_adv
```

bufr_filter reminder

To run bufr_filter we need a BUFR file and a filter (text file with bufr_filter rules) and we may produce an output BUFR file, or just print some data.

```
bufr_filter –o output.bufr filter_text_file input.bufr
```



BUFR input file

bufr_filter

```
# filter rules
set unpack=1;
print “latitude=[latitude]”;
```

BUFR output file

bufr_filter to inspect files

1. Inspect the contents of a BUFR file. Create a filter file and use it to print the following keys for the **synop1.bufr** file:

   unexpandedDescriptors

   expandedDescriptors

   expandedAbbreviations

   expandedNames

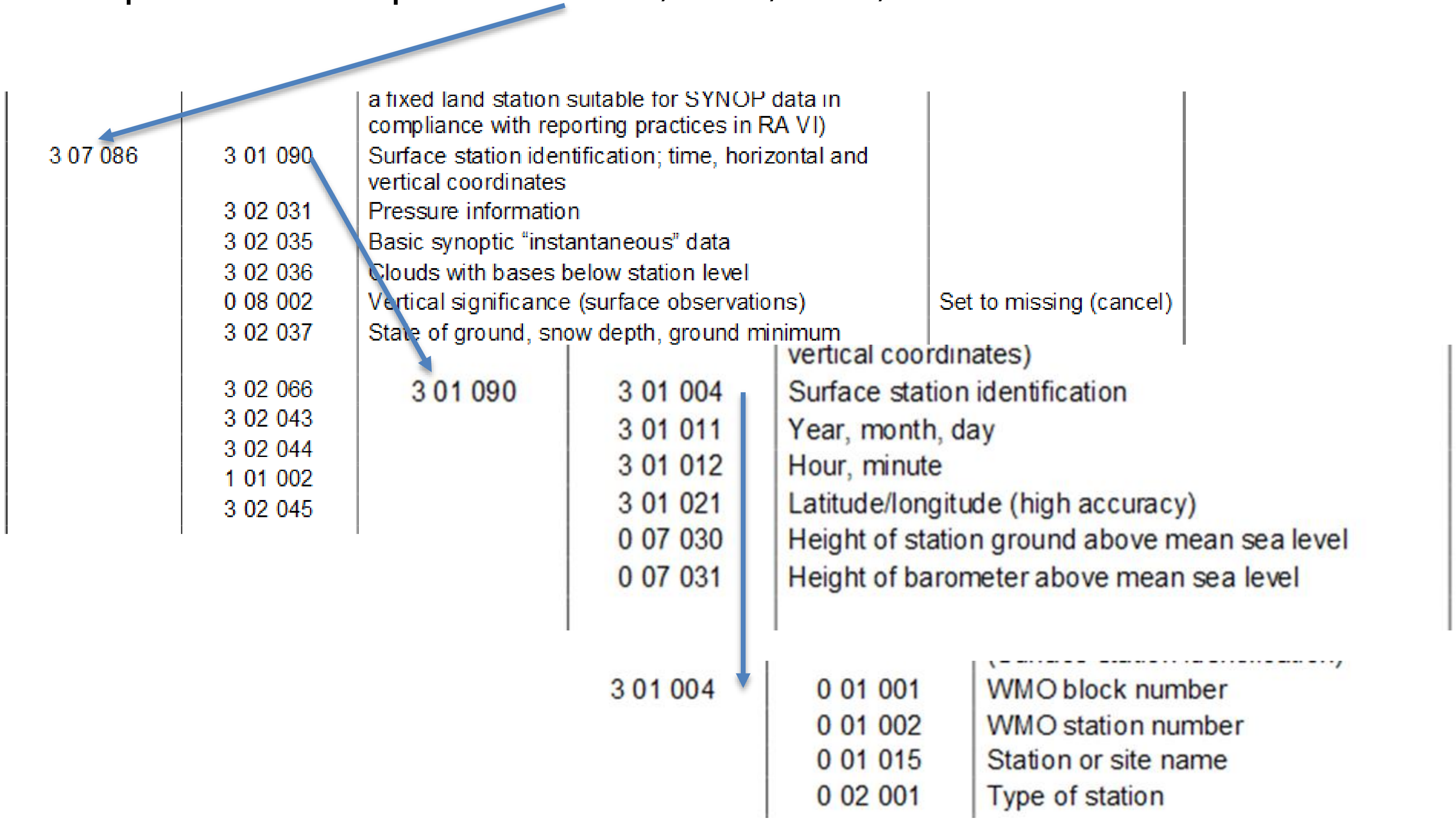
2. Print the **latitude,longitude,airTemperature** data (what happens if you don’t set unpack=1 in your filter?).
3. For **airTemperature**, print the **units,code,width,reference** and **scale** (which are the attributes of the key airTemperature) you can access them via the operator “→”.

Solution exercise 1

```
# usage :
#   bufr_filter pex1.flt synop1.bufr
#############################################

print " unexpandedDescriptors=[unexpandedDescriptors!1,]";
print " expandedDescriptors=[expandedDescriptors!1,]";
print " expandedAbbreviations=[expandedAbbreviations!1,]";
print " expandedNames=[expandedNames!1,]";

##################################################
# to access data section we must unpack the data
##################################################
set unpack=1;

print " latitude=[latitude]";
print " longitude=[longitude]";
print " airTemperature=[airTemperature]";

#################################################
# to access the attributes we use -> operator
################################################

print "airTemperature units = [airTemperature->units]";
print "airTemperature code  = [airTemperature->code]";
print "airTemperature width = [airTemperature->width]";
print "airTemperature scale = [airTemperature->scale]";
print "airTemperature ref   = [airTemperature->reference]";
```

Expansion process

- unexpandedDescriptors=307086,5001,6001,7001



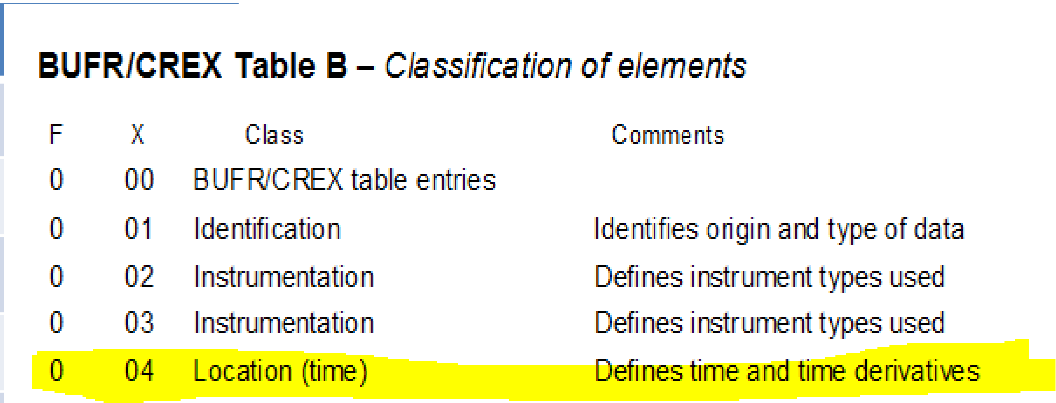
		a fixed land station suitable for SYNOP data in compliance with reporting practices in RA VI)	
3 07 086	3 01 090	Surface station identification; time, horizontal and vertical coordinates	
	3 02 031	Pressure information	
	3 02 035	Basic synoptic "instantaneous" data	
	3 02 036	Clouds with bases below station level	
	0 08 002	Vertical significance (surface observations)	Set to missing (cancel)
	3 02 037	State of ground, snow depth, ground minimum	
	3 02 066		
	3 02 043		
	3 02 044		
	1 01 002		
	3 02 045		

		vertical coordinates)
3 01 090	3 01 004	Surface station identification
	3 01 011	Year, month, day
	3 01 012	Hour, minute
	3 01 021	Latitude/longitude (high accuracy)
	0 07 030	Height of station ground above mean sea level
	0 07 031	Height of barometer above mean sea level

3 01 004	0 01 001	WMO block number
	0 01 002	WMO station number
	0 01 015	Station or site name
	0 02 001	Type of station

Expansion process

1001	WMO BLOCK NUMBER
1002	WMO STATION NUMBER
2001	TYPE OF STATION
4001	YEAR
4002	MONTH



BUFR/CREX Table B – ***Classification of elements***

F	X	Class	Comments
0	00	BUFR/CREX table entries	
0	01	Identification	Identifies origin and type of data
0	02	Instrumentation	Defines instrument types used
0	03	Instrumentation	Defines instrument types used
0	04	Location (time)	Defines time and time derivatives

This process of expansion begins with each of the unexpandedDescriptors and expands each one till we have the “atomic” descriptors for stationNumber, year, etc.

Notice that the descriptors 004001,004002 are related to Time,001 with Identification, 002 with type of instrument etc.

Unpack

We have to set unpack=1 to access the data section in the BUFR message, otherwise no data can be read from the data section.

However, we may read the header section without using unpack=1.

The **airTemperature** has attributes, such as units, scale etc that can be accessed by the arrow operator. Again, we have to set unpack=1 to access the attributes of a variable.

bufr_filter to inspect files

Use **bufr_filter** to print the attribute **percentConfidence** of the key **pressureReducedToMeanSeaLevel** from the file:

synop_with_confidence.bufr

Print all the attributes of **pressureReducedToMeanSeaLevel**.

Hint use **bufr_dump** with the option **–jf** to display all attributes, **grep** can also be handy to filter out the output.

Solution exercise 2

With

bufr_dump –jf synop_with_confidence.bufr

we access all attributes

```
  "key" : "airTemperatureAt2M",
  "value" : 302.7,
  "index" : 18,
  "code" : "012004",
  "units" : "K",
  "scale" : 1,
  "reference" : 0,
  "width" : 12
},
```

Solution exercise 2

```
# usage
#   bufr_filter pex2.flt synop_with_confidence.bufr
####################################################

# to access the data section unpack is needed
set unpack=1;

print " pressureReducedToMeanSeaLevel=[pressureReducedToMeanSeaLevel]";
print " pressureReducedToMeanSeaLevel->percentConfidence=[pressureReducedToMeanSeaLevel->percentConfidence]";

#### print all the attrbutes


print " pressureReducedToMeanSeaLevel->code=[pressureReducedToMeanSeaLevel->code]";
print " pressureReducedToMeanSeaLevel->index=[pressureReducedToMeanSeaLevel->index]";
print " pressureReducedToMeanSeaLevel->units=[pressureReducedToMeanSeaLevel->units]";
print " pressureReducedToMeanSeaLevel->reference=[pressureReducedToMeanSeaLevel->reference]";
print " pressureReducedToMeanSeaLevel->scale=[pressureReducedToMeanSeaLevel->scale]";
```

bufr_filter access by rank/condtion(exercise 3)

Print **latitude,longitude,airTemperature** from the file **temp.bufr**.

Get a dump of the file in JSON and compare it with the result of the **bufr_filter**.

By using **bufr_filter** and accessing keys by rank find the second **windSpeed** and **windDirection**.

By using **access by condition**(Using /key=value/ syntax), print **latitude,longitude,airTemperature** for specific geopotential levels (for example **nonCoordinateGeopotentialHeight** value of 1035 gpm.).

Solution exercise 3

We can combine bufr_dump and other Linux tools

```
bufr_dump -ja temp.bufr|grep -c airTemperature
```

Gives you the number of airTemperature keys in the file.

```
bufr_dump -ja temp.bufr | grep -C1 "airTemperature"  |grep "value"
```

Combined with the output of bufr_filter we can access the keys by rank

```
print "second windspeed=[#2#windSpeed]";

print "second windDirection=[#2#windDirection]";
```

Or by condition

```
print "/nonCoordinateGeopotentialHeight=1035/airTemperature=

[/nonCoordinateGeopotentialHeight=1035/airTemperature]";
```

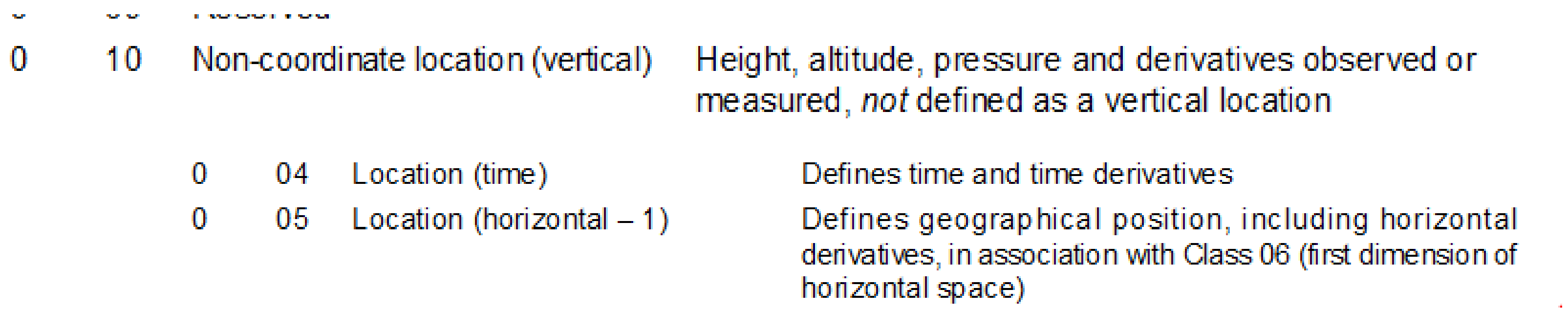
Solution for temp.bufr exercise 3

```
set unpack=1;
#getting all airTemperatures in 4 (!4) columns.
#Each column contains 8 digit wide numbers with 2 decimal places comma separated
print "latitude=[latitude!4]";
print "longitude=[longitude!4]";
print "airTemperature=[airTemperature!4%8.2f',']";
# getting values by rank  (second windSpeed and windDirection)
print "#2#windSpeed=[#2#windSpeed]";
print "#2#windDirection=[#2#windDirection]";
# getting values with a condition
print "for pressure=66850  windDirection=[/pressure=66850/windDirection]";
print "for pressure=66850  windSpeed=[/pressure=66850/windSpeed]";
#getting the airTemperature at geoPotential 1035
print
"/nonCoordinateGeopotentialHeight=1035/airTemperature=[/nonCoordinateGeopotentialHei
ght=1035/airTemperature]";
print "/nonCoordinateGeopotentialHeight=1035/airTemperature-
>units=[/nonCoordinateGeopotentialHeight=1035/airTemperature->units]";
```

nonCoordinate

```
"key" : "nonCoordinateGeopotentialHeight",
"value" : 35,
"index" : 33,
"code" : "010009",
"units" : "gpm",
"scale" : 0,
"reference" : -1000,
"width" : 17

},
```

This **nonCoordinateGeopotentialHeight** (code 010009) means that this is an observed value by itself, opposite to other descriptors in Table B (0 04YYY , etc) that are coordinates and so identify an observation.



0	10	Non-coordinate location (vertical)	Height, altitude, pressure and derivatives observed or measured, *not* defined as a vertical location

0	04	Location (time)	Defines time and time derivatives
0	05	Location (horizontal – 1)	Defines geographical position, including horizontal derivatives, in association with Class 06 (first dimension of horizontal space)

bufr_filter to inspect uncompressed data(exercise 4)

BUFR can work with uncompressed data.

Use the file **synop_multi_subset.bufr** to print the following information

- **compressedData**.
- **stationNumber,stationOrSiteName,latitude,longitude,airTemperature,dewPointTemperature** for subsets number 3 and 5.

Check your results with the JSON output of **bufr_dump** and **bufr_dump -p**.

Solution inspect uncompressed data exercise 4

```
###############################################
# print several keys for uncompressed data
# file synop_multi_subset.bufr
##############################################
set unpack=1;
print "compressedData=[compressedData]";
print "stationNumber  = [/subsetNumber=3/stationNumber]";
print "stationName  = [/subsetNumber=3/stationOrSiteName]";
print "latitude  = [/subsetNumber=3/latitude]";
print "longitude  = [/subsetNumber=3/longitude]";
print "airTemperature  = [/subsetNumber=3/airTemperature]";
print "dewPointTemperature  = [/subsetNumber=3/dewpointTemperature]";
print ;
print "stationNumber  = [/subsetNumber=5/stationNumber]";
print "stationOrSiteName  = [/subsetNumber=5/stationOrSiteName]";
print "latitude  = [/subsetNumber=5/latitude]";
print "longitude  = [/subsetNumber=5/longitude]";
print "airTemperature  = [/subsetNumber=5/airTemperature]";
print "dewPointTemperature  = [/subsetNumber=5/dewpointTemperature]";
```

bufr_filter to inspect compressed data(exercise 5)

BUFR files can contain compressed data, **bufr_filter** works also with compressed data.

Use the file **scatterometer.bufr** to retrieve:

- **compressedData** key.
- **latitude,longitude** keys.
- the **backscatter** for all subsets with **beamIdentifier 2**. You can use some formatting to improve the output.

Hint: Refer back to the **bufr_filter** introduction slides.

Check your results with the output of **bufr_dump,** the option **–p** may be handy.

Solution compressed data exercise 5

```
set unpack=1;

print "compressedData=[compressedData]";

print "latitude=[latitude!5%.2f',']";

print "longitude=[longitude!5%.2f',']";

print "backscatter=[/beamIdentifier=2/backscatter!6%.2f]";
```

bufr_filter to extract subsets

Sometimes we may need to retrieve only a small number of subsets. To do so we have to set some keys.

To extract only the fourth subset

```
set unpack=1;
set extractSubset=4;
set doExtractSubsets=1;
write;
```

To extract from subset 3 to subset 10

```
set unpack=1;
set extractSubsetIntervalStart=3;
set extractSubsetIntervalEnd=10;
set doExtractSubsets=1;
write;
```

Remark: Remember to ***set doExtractSubsets*** to 1 before writing. This filtering works for **compressed** and **uncompressed** data.

bufr_filter to extract subsets in an area

bufr_filter can be used to extract subsets within an area defined by a bounding box.

This feature only works for **compressed data.**

This may be useful with satellite data, to retrieve all the subsets in a certain area defined by a bounding box.

bufr_filter to extract subsets in an area

```
set unpack=1;
transient originalNumberOfSubsets=numberOfSubsets;
set extractAreaNorthLatitude=26.0;
set extractAreaSouthLatitude=23.0;
set extractAreaEastLongitude=-35.0;
set extractAreaWestLongitude=-55.0;

set doExtractArea=1;

if ( extractedAreaNumberOfSubsets != 0) {
   write;
}
print "extracted [extractedAreaNumberOfSubsets] of [originalNumberOfSubsets]
subsets";
```

bufr_filter to extract subsets in an area

Comments

The bounding box of the region must be set, through the keys

extractAreaNorthLatitude,
extractAreaSouthLatitude,
extractAreaEastLongititude
extractAreaWestLongitude

The if statement avoids writing if the extraction does not find any subsets in the given area.

A ***transient*** variable is used to keep track of the number of subsets in the message and see how many have been selected (through the key extractedAreaNumberOfSubsets).

bufr_filter to extract subsets in an area

Using the file **amv2_87.bufr** and **bufr_filter** do an area extraction for a relevant area.

Use **bufr_dump** to see the ranges of latitudes and longitudes.

How many subsets were selected?

Solution area extraction exercise 6

```
#################################################
# extracts an area defined from the bounding box
# file amv2_87.bufr
##################################################
set unpack=1;
transient originalNumberOfSubsets=numberOfSubsets;

set extractAreaNorthLatitude=26.0;
set extractAreaSouthLatitude=23.0;
set extractAreaEastLongitude=-35.0;
set extractAreaWestLongitude=-55.0;

set doExtractArea=1;

if ( extractedAreaNumberOfSubsets != 0) {
    write;
}
print "extracted [extractedAreaNumberOfSubsets] of [originalNumberOfSubsets]
subsets";
```

bufr_filter to extract subsets in a time range

To select subsets in a given time range the following keys are provided:

```
extractDateTimeYearStart
extractDateTimeMonthStart
extractDateTimeDayStart
extractDateTimeHourStart
extractDateTimeMinuteStart
extractDateTimeSecondStart
extractDateTimeYearEnd
extractDateTimeMonthEnd
extractDateTimeDayEnd
extractDateTimeHourEnd
extractDateTimeMinuteEnd
extractDateTimeSecondEnd
extractedDateTimeNumberOfSubsets
doExtractDateTime   must be set to 1 to  extract
```

bufr_filter to extract subsets in a time range

To do the actual selection don’t forget to use

```
set doExtractDateTime=1;
```

NOTES

The full start time and the full end time must be specified to actually do the extraction.

As with area selection, this feature works only with **compressed data**.

bufr_filter to extract subsets in a time range

Using the file scatterometer.bufr and **bufr_filter**, get the number of subsets where seconds is in the interval [26,30]. How many subsets do you get? Write the result into a file, and using **bufr_dump** check that the subsets are within the time interval.

Solution time extraction exercise 7

```
################################################
#extracts the subsets within a time range
#file : scatterometer.bufr
#################################################
set unpack=1;
print "second=[second!7%0d' ']";
#define start YMDHMS
set extractDateTimeYearStart=2012;
set extractDateTimeMonthStart=11;
set extractDateTimeDayStart=2;
set extractDateTimeHourStart=0;
set extractDateTimeMinuteStart=24;
set extractDateTimeSecondStart=26;
#define end YMDHMS
set extractDateTimeYearEnd=2012;
set extractDateTimeMonthEnd=11;
set extractDateTimeDayEnd=2;
set extractDateTimeHourEnd=0;
set extractDateTimeMinuteEnd=24;
set extractDateTimeSecondEnd=30;
set doExtractDateTime=1;
print "number of extractedSubsets =[extractedDateTimeNumberOfSubsets]";
if ( extractedDateTimeNumberOfSubsets >0 ) {
   write;
}
```

bufr_filter for simple thinning

With high resolution data, it may be needed to reduce the number of observations.

To allow thinning, the following keys are provided.

- **simpleThinningSkip**
- **doSimpleThinning**

```
# allows thinning of the BUFR file
set unpack=1;
set simpleThinningSkip=5; # take  subsets 1,7,13
set doSimpleThinning=1;  # does the actual thinning
set pack=1;
write;
```

bufr_filter for simple thinning

Use the file scatterometer.bufr and **bufr_filter** to thin the observations taking observations 1,7,13. Check your results with **bufr_dump**.

How many subsets do you have in the original file? And in the “thinned” file?

Solution exercise 8

```
###########################################
# does the thinning of observations
# file : scatterometer.bufr
##########################################
set unpack=1;
set simpleThinningSkip=5;
set doSimpleThinning=1;
set pack=1;
write;
```

creating BUFR messages with bufr_filter

Using **bufr_filter** we can create new messages. To do so, we need:

- An input BUFR file.
- Set the compressedData flag to 1→for compressed data or 0→ uncompressed data
- Set the key **unexpandedDescriptors** to the list of descriptors of the new message.

creating BUFR messages with bufr_filter

Remarks

When we create a new message with **bufr_filter** by setting the **unexpandedDescriptors**, the library is using the input message only as a seed, to select the proper tables etc.

Once we set **unexpandedDescriptors** the section 3 is **fully set**. We can not change any key in section 3 after setting the **unexpandedDescriptors**. **The keys that intend to change the structure of the message i.e. section 3(compressedData, delayedDescriptorReplicationFactor, bit maps etc.) should be set BEFORE setting unexpanded descriptors**.

If the input message has a very old table version (11 for example) the setting of **unexpandedDescriptors** doesn't work.

creating BUFR messages with bufr_filter(exercise 9a)

We can use **bufr_filter** to create a new BUFR message by setting the values of some keys. In particular, the **unexpandedDescriptors** keys allows us to create a new BUFR message.

1.-Set the **unexpandedDescriptors** key to the following list

{106002, 008002 ,104003 ,005002, 006002 ,010002, 012001}

As an input file you can use **synop.bufr**. This input file is used as a seed, to select the proper tables.

By setting the key **unexpandedDescriptors** we are actually creating a new message with all the key values set to MISSING (null).

2.-Check the original and the newly created messages with **bufr_dump**.

3.-Print the following keys for the new message :

unexpandedDescriptors, expandedDescriptors, expandedAbbreviations

Solution exercise 9a

```
##################################################

# creates a message from the unexpandedDescriptors

##################################################

set unpack=1;

set unexpandedDescriptors={106002, 008002, 104003, 005002, 006002,
010002,012001};

set pack=1;

write;
```

Creating BUFR messages with bufr_filter (exercise 9b)

- Set some values to the message created in the previous practical for the second instance of **latitude, longitude, nonCoordinateHeight** and **airTemperature**.

- How many subsets do you have in your file?

Solution exercise 9b continues

```
set unpack=1;

set unexpandedDescriptors={106002, 008002, 104003, 005002, 006002,
010002,012001};

set #2#latitude=15;

set #2#longitude=25;

set #2#nonCoordinateHeight=1000;

set #2#airTemperature=219.25;

set pack=1;

write;
```

What if(exercise 10)

Sometimes we have to deal with old messages, or messages that use old tables. In this case, we may not get the right results.

Using the file **old_amdar.bufr** as an input and **bufr_filter**, try to create a new message with the following ***unexpandedDescriptors***:

{106002, 008002, 104003, 005002, 006002, 010002,012001}

Does it work? What happened? Check with **bufr_dump** the old_amdar.bufr file.

Solution exercise 10

```
bufr_filter -o wrong.bufr creating.filter  old_amdar.bufr
```

```
ECCODES ERROR   :  unable to find definition file sequence.def in
bufr/tables/11/wmo/11/sequence.def::bufr/tables/11/local/1/21/0/sequence.def

Definition files
path="/usr/local/apps/eccodes/2.6.0/GNU/5.3.0/share/eccodes/definitions"

ECCODES ERROR   :  unable to get hash value for sequences

ECCODES ERROR   :  Error while setting key unpack (Hash array no match)

ERROR: Hash array no match
```

Solution ex 10

This Tables problem is due to the use of old tables (Version 11) at the originating centre. This can be corrected, by creating a filter that changes the header information, (MasterTablesVersionNumber, localTablesVersionNumber etc)

```
set masterTableNumber=0;
set masterTablesVersionNumber=13;
set localTablesVersionNumber=1;
set bufrHeaderCentre=21;
write;
```

So if we use this rules file to modify the old_amdar.bufr we get an usable BUFR file again (new_amdar.bufr)

```
bufr_filter -o new_amdar.bufr changeTables_back.filter old_amdar.bufr
```