RFC: Code Refactoring for h5dump

Peter Cao

This RFC describes the work of code refactoring for h5dump.

1 Introduction

Maintainable and extendable code is an important part of our work toward high quality software. We started an RFC on structural layout design for a tool library[1] a year ago. This RFC will focus on improving the source code of h5dump.

Better code structure and design has many benefits. Below we list a few:

- More maintainable: easier to understand, fix bugs in and test
- More extendable: easier to modify and add new features

We start with h5dump for three reasons: 1) h5dump is one of the most used tools; 2) h5dump source code is complicated and unorganized; 3) we have an urgent need to merge the source code for showing packed-bits to the current source repository. Before adding the code to h5dump, we need a better structure for the current h5dump source code.

Restructuring the code means

- moving code to separate files based on features;
- breaking up large functions into small functions;
- renaming functions and variables for better consistency;
- fixing defects such as dead code, redundant code, warnings, and other defects;
- replacing algorithms or conditions for better performance;
- modifying or adding tests, and
- improving documentation.

However, these modifications will not

- change the functionality and user interface for h5dump,
- rewrite the h5dump tool, which is an accumulation of over ten years of development work.

[1] https://www.hdfgroup.uiuc.edu/RFC/HDF5/tools/general/RFC_Tools_Lib_v1.pdf

2 Code assessment

h5dump is one of the command-line utilities developed with the first release of the HDF5 library in 1998. Over the years many features and options were added to h5dump and the source code became larger and poorly organized. Debugging defects or adding any new features has become very difficult since the code is not well designed and organized.

The following table is a list of the source files that are used by h5dump (in tools/h5dump directory). h5dump also uses functions in the tools library. This work does not include reconstructing of the tools library, which has about 20,000 lines of code.

Table 1 -- h5dump source files

File	Lines of Code	Description
h5dump.c	7,175	Main source code for h5dump
h5dumpgentest.c	7,161	Generating testing files
testh5dump.sh.in	646	Testing scripts
testh5dumpxml.sh.in	201	Scripts for testing xml output
h5dump.h	160	Header file for h5dump
Total	15,343	

3 Tasks and work estimation

The first priority is to reconstruct the main h5dump code, h5dump.c, which have more 7,000 lines of code. We also continue to fix defects in h5dump tool as part of Coverity session efforts according to the coverity assessment information.

3.1 Add code for packed bits (16 hours)

The feature for printing values of packed bits was added to the HDF5 1.8 branch by using a macro, H5_HAVE_H5DUMP_PACKED_BITS. The code is not in the HDF5 trunk. This task is to have this feature in both HDF5 trunk and 1.8 branches using better organized structure. Since this part of the code is relatively independent and small, it should not be so hard to reorganize the code using the feature template discussed above.

3.2 Remove dead code and repeated code (40 hours)

Going through the current code and removing dead code. There are also some code functions that have already implemented in the tools library. We will remove the repeated code and use the implementation from tools library.

3.3 Separate the XML feature (16 hours)

The part for handling xml has about 2000 lines of code. This task is to move the xml feature to a separate file. The major functions for writing xml include:

- xml_dump_group,
- xml_dump_named_datatype,

- xml_dump_dataset,
- xml_dump_dataspace,
- xml_dump_datatype,
- xml_dump_attr,
- xml_dump_data

3.4 Revise feature functions (32 hours)

The first task is to make independent features into easily maintainable components. The main code frame will just hold a list of features functions and the implementation of individual features will be moved to separate files. Some of the separate features in h5dump include:

- Write to xml format,
- Print header information only,
- Dump groups,
- Dump datasets,
- Dump named datatypes,
- Show packed bits,
- Print content list (a list of objects in a file),
- and etc.

Using an easily maintained template is not new. There is already a global dispatch table for the dump functions. We will revise the list and move the implementation to separate files as needed.

```
typedef struct dump_functions_t {
   void    (*dump_group_function) (hid_t, const char *);
   void    (*dump_named_datatype_function) (hid_t, const char *);
   void    (*dump_dataset_function) (hid_t, const char *, struct subset_t *);
   void    (*dump_dataspace_function) (hid_t);
   void    (*dump_datatype_function) (hid_t);
   herr_t  (*dump_attribute_function) (hid_t, const char *, const H5A_info_t *, void *);
   void    (*dump_data_function) (hid_t, int, struct subset_t *, int);
} dump_functions;
```

3.5 Identify other defects

We will go through h5dunmp code to identify other defects and fix them according their priorities. Examples of other defects include:

- Poor performance, e.g. calling datatype information on each data value not on dataset level
- Unnecessary memory use
- Dead code or repeated code (as mentioned above)

- Inconsistent user interface
- Incomplete tests
- Complicated function calls
- No or incomplete documentation/comments on code/functions
- etc

Revision History

March 25, 2011:	Version 1 circulated for comment within The HDF Group.
May 5, 2011:	Version 2 revised after the technical discussion with the group.