RFC: h5edit – An HDF5 File Editing Tool

Albert Cheng, Jonathan Kim, Elena Pourmal

This document proposes a new tool for editing an HDF5 file and solicits requirements for the tool's functionality and interface.

1 Introduction

The HDF5 distribution contains command line utilities for creating datasets (h5import, h5copy) and groups (h5mkgrp, h5copy). Those tools become handy, when a modification like adding new dataset or new group to an existing file or collection of files has to be done without actual programming, or when a new file is created while one works on a data layout prototype in HDF5 file. None of the utilities mentioned above, can create HDF5 attributes, and therefore one has to write and run a program to do so.

To address the lack of such functionality in HDF5, we propose to create an HDF5 file-editing tool called, `h5edit`. The initial version provides the ability to handle attributes in an HDF5 file. This document outlines the major use cases, discusses examples of tool interface, and solicits requirements for the tool.

2 Use cases for handling attributes

This section describes several use cases that illustrate how the users may use the tool to handle attributes in an HDF5 file.

- A user would like to add an attribute to an HDF5 object (e.g., dataset or group) without writing a program. He uses the `h5edit` tool and specifies necessary input on a command line to add the attribute.
- A user would like to use a visualization tool to display some data stored in an HDF5 dataset, but the tool requires an attribute with a special name on the dataset in order to display the data. The user uses `h5edit` to add the required attribute and proceeds with the visualization.
- A user runs an application in a parallel environment to create a big HDF5 file with size of multiple gigabytes, containing thousands of datasets. When he runs a different application to read the data, he discovers that he has made a mistake and used the wrong name for an attribute in each of the datasets, making his file incompatible with the second application. Instead of rewriting and rerunning his first application, he uses the `h5edit` tool to delete the incorrectly named attribute from all datasets and add a new one with the correct name to every dataset.

- A user wants to add an attribute with a complex data type such as an array of nested C-language structure. The user creates an input file with a description of the data type along with other required information for the attribute. The user uses `h5edit` to read the input file to generate the attributes.
- A user wants to add several attributes to a dataset. The user creates an input file containing the description of the attributes. He specifies the name of the input file along with the name of the dataset as input to `h5edit` to generate the attribute.
- A user has a collection of HDF5 files and would like to add (or remove, or copy) an attribute to a group in every file. User creates a shell script and uses `h5edit` to modify the files.
- A user wants to duplicate an attribute in a new HDF5 file, as it is in an original file. The user runs `h5dump` on the original HDF5 file and finds HDF5 DDL (http://www.hdfgroup.org/HDF5/doc/ddl.html) representation of the desired attribute. The user copies the attribute's DDL description to an input file and makes necessary changes. The user specifies the input file as an input to `h5edit`.
- A user would like to change an attribute value. He uses `h5edit` and specifies the name of the attribute and its new value on the command line to change the value stored in an HDF5 file.

3 Proposal for the h5edit tool

To address the user cases described above, we propose to implement the `h5edit` tool to perform operations on HDF5 objects in an HDF5 file. In the following, we will list out the proposed functionality, editing commands and options of the h5edit tool.

3.1 Functionality

Sections below discuss the command syntax for creating and deleting attributes since those are given a high priority by the HDF5 users and will be implemented in the initial version of the tool.

The initial version of h5edit supports attributes related operations. Operations for other objects such as datasets and groups will be added in future versions. Attributes operations are listed as follows:

- `create` : create a new attribute in an object
- `delete`: delete an existing attribute from an object

Other operations to consider are:

- `rename:` modify the name of an existing attribute
- `modify:` modify the value(s) of an existing attribute
- `copy:` copy an existing attribute from one object to another object
- `move:` move an existing attribute from one object to another object
- `exists:` check if an attribute exists

Suggestions for other operations are more than welcome.

Sections below discuss the command syntax for creating and deleting attributes since those are given a high priority by the HDF5 users and will be implemented in the first version of the h5edit tool.

3.2 H5edit Command Language

H5edit commands are in the form of

<command> <command-parameters> ;

<command> is keyword based and indicates the operation desired. *<command-parameters>* specifies information required to perform the operation. The semicolon indicates the end of the command. A formal definition of the h5edit Command Language is presented in *Appendix 1: Definition of H5edit Command Language.*

3.2.1 Creating attributes

Syntax:

- create ATTIBUTE *ATTRIBUTE_NAME ATTRIBUTE_VALUE* [*OBJECT_NAME* …] ;

Description:

This command creates an attribute with a name specified by *ATTRIBUTE_NAME,* according to the value specified by the parameter *ATTRIBUTE_VALUE*, in objects specified by one or more *OBJECT_NAME.*

The *ATTRIBUTE_VALUE* consists of three parts, *attribute_type, attribute_space,* and *attribute_data.* The *attribute_type* and *attribute_space* parts are optional.

The *attribute_type* specifies the data type[1] of the attribute. If it is missing, the default value of H5T_NATIVE_INT (C native int) is used.

The *attribute_space* specifies the data space of the attribute. If it is missing, the *attribute_space* is defined according to *attribute_data* given. If only one data point is given, *attribute_space* is defined as H5S_SCALAR (scalar dataspace). If more than one data points are given, *attribute_space* is defined as a 1-D array with the dimension size equals to the number of data points given.

Examples:

The following three h5edit commands show how to create an integer, a pair of floating points, and a string attributes in the objects */m1 and /m2*.

```
create ATTRIBUTE Percentage_per_Volume 40 /m1;
create ATTRIBUTE GPS_Location {
      DATATYPE H5T_IEEE_F32LE
      DATASPACE SIMPLE {(2)/(2)}
      DATA {0.0, 180.0}
      }
      /m2;
create ATTRIBUTE “Temp Scale” {
      DATATYPE H5T_C_S1
      DATA {“Celsius”}
```

[1] http://www.hdfgroup.org/HDF5/doc/RM/PredefDTypes.html shows the predefined types

```
}
/m1 /m2;
```

The first command creates a scalar attribute Percentage_per_Volume of native int datatype with the value of 40 in object /m1.

The second command creates an attribute GPS_Location that is a 1-D array of two 32bits floating point datatype, with the values as (0.0, 180.0), in the object /m2.

The last example creates a scalar attribute "Temp Scale" of string datatype, with the value as "Celsius", in each of the two objects, `/m1` and `/m2`. `Note that the attribute name needs to be quoted if it contains white spaces such as the blank.`

The keywords, DATATYPE, DATASPACE and DATA are optional. They are used for clarity and can be omitted if no ambiguities arise.

3.2.2 Deleting attributes

Syntax:

- `delete ATTIBUTE` *`ATTRIBUTE_NAME`* `[`*`OBJECT_NAME`* `…] ;`

Description:

This command deletes an existing attribute with a name specified by *ATTRIBUTE_NAME*, from the objects specified by one or more *OBJECT_NAME.*

Examples:

The following commands show how to delete attributes.

```
delete ATTRIBUTE Percentage_per_Volume /m1;
delete ATTRIBUTE "Temp Scale" /m1 /m2;
```

The first command deletes the attribute *Percentage_per_Volume* from the object */m1.* The second command deletes the attribute "*Temp Scale"* from both objects */m1* and */m2.*

3.3 H5edit Tool Options

Syntax:

- `h5edit` *`<option>`* `[`*`<option-parameter>`*`]` *`<h5file>`*

Description:

<h5file> is the HDF5 file to be edited. *<option>* is one of the tool options as follows:

--command-file *command-file-name*

The file *command-file-name* contains h5edit commands that are to be applied to the HDF5 file.

--command *edit-commands*

One or more h5edit commands, separated by comma, to be applied to the HDF5 file. This is meant for some simple h5edit commands. For more complex commands, the user should use the –command-file option.

--dryrun

Just check the syntax of the h5edit commands against the HDF5 file without making the actual changes to the HDF5 file.

Examples:

- `h5edit --command "delete ATTRIBUTE Percentage_per_Volume /m1; create ATTRIBUTE Percentage_per_Volume 40 /m1;" file1.h5`

This effectively changes the value of the attribute `Percentage_per_Volume` of object `/m1` to the integer value of `40`.

- `h5edit --command-file change-cmds file1.h5`

The contents of the file *change-cmds* can be like the following and they effective change the values of attribute "Temp Scale" of objects /m1, /m2, …, /m10 to the value of "Celsius".

```
delete ATTRIBUTE "Temp Scale" /m1 /m2 /m3 /m4 /m5 /m6 /m7 /m8 /m9 /m10;
create ATTRIBUTE "Temp Scale" {
      DATATYPE H5T_C_S1
      DATA {"Celsius"}
      }
      /m1 /m2 /m3 /m4 /m5 /m6 /m7 /m8 /m9 /m10;
```

As one can see, attempt to do these changes via the --command option, though possible, would be messy and hard to follow.

4 Atomicity of h5edit

The operation effect of h5edit should be atomic, that is, the changes must be done as all or nothing. Consider a case, similar to the above example, but it tries to change the attribute "`Temp_Scale`" values of objects `/m1, /m2, …, /m10.`

It happens that every specified object, except /m50, has the attribute "Temp Scale". Therefore, the h5edit command will fail halfway through, with /m1 to /m49 changed. Not all users like to have the data file partially changed. They would rather have no changes applied to the original data file, find out the cause of the failure, correct the commands and apply them again without errors.

It is also crucial that h5edit should avoid exiting the program with the HDF5 data file in an unstable state. Consider the example above, when h5edit detects that it cannot delete the attribute from /m50, it should print an error message explaining the cause of the failure, close all open objects, close the data file, then exit with a failure code.

4.1 No Atomicity Option

A naïve way to support the Atomicity of the h5edit tool is to make a backup copy of the HDF5 data file at the very beginning of h5edit and then processes the commands requested. If it succeeds, discards the backup copy and exits. If it fails somewhere, discards the changed file and restores the original file from the backup copy. This is expensive and some users may not care. Therefore, h5edit will support atomic change as the default but also provide an option, say, --noatomic, to allow partial changes to occur. Again, the partially changed data file must be a properly closed HDF5 file.

4.2 Incremental Atomicity Option

Some users may prefer an option of --inc-atomic for incremental atomicity which means atomicity at individual command level is desired, not the entire execution. A simple but not necessarily most efficient way to implement the *incremental atomicity* is to close the data file after every successful execution of each command, save a backup copy of the data file, reopen the file for the next command. If the next command fails, the data file is restored from the backup copy.

4.3 Summary of Atomicity Options

--atomic (default)

The changes must be done as all or nothing. The original data file is restored in case of any command failures.

--no-atomic

No atomicity is desired. Do as much changes as possible.

--inc-atomic

Atomicity of changes at individual command level is desired, not the entire execution.

5 Exit code from the h5edit

- 0: when the operation succeeds.
- 1: when the operation fails.

However more error conditions can be added as work progresses. If added, the exit code values will be adjusted accordingly. In such case, the exit code values will be bigger than 0.

6 Conclusion

Currently a user has to write a program to add or remove attributes from objects in the HDF5 file. It is inconvenient to deal with programming repeatedly when the attribute is simple or the operation over the attribute involves simple actions.

The `h5edit` tool will provide a convenient way to manipulate objects in the HDF5 file. For the initial implementation, essential operations like creating and deleting attributes will be provided. Additional functionality can be added in future versions.

Appendix 1: Definition of H5edit Command Language

1. Introduction

This section describes the command language (CL) of the *h5edit* tool. The description is in Backus-Naur Form.

2. Explanation of Symbols

This section contains a brief explanation of the symbols used in the CL.

```
    ::=                         defined as
    <tname>                     a token with the name tname
    <a> | <b>                   one of <a> or <b>
    <a>opt                      zero or one occurrence of <a>
    <a>*                        zero or more occurrence of <a>
    <a>+                        one or more occurrence of <a>
    [0-9]                       an element in the range between 0 and 9
    `['                         the token within the quotes (used for special
characters)
    TBD                         To Be Decided
```

3. The h5edit CL

```
<h5edit_command_file> ::= <h5edit_statement>+

<h5edit_statement> ::= <h5edit_command> ;

<h5edit_command> ::= <Attribute_create_command> | <attribute_delete_command>

<attribute_create_command> ::= create <attribute_name> <attribute_definition>
        <object_name>+

<attribute_name> ::= ATTRIBUTE <name>

<object_name> ::= <group_name> | <dataset_name>

<group_name> ::= GROUP opt <name>

<dataset_name> ::= DATASET opt <name>

<attribute_definition> ::= { <attribute_datatype_definition> opt
        <attribute_dataspace_definition> opt <attribute_data> }

<attribute_datatype_definition> ::= DATATYPE opt <datatype_definition>

<attribute_dataspace_definition> ::= DATASPACE opt <dataspace_definition>

<attribute_data> ::= DATA opt { <data> , <data>* }

<attribute_delete_command> ::= delete <attribute_name> <object_name>+
```

4. Examples

```
create ATTRIBUTE Percentage_per_Volume 40 /m1;

create ATTRIBUTE GPS_Location {
      DATATYPE H5T_IEEE_F32LE
      DATASPACE SIMPLE {(2)/(2)}
      DATA {0.0, 180.0}
      }
      /m2;

delete ATTRIBUTE “Temp Scale” /m1 /m2 /m3 /m4 /m5 /m6 /m7 /m8 /m9 /m10;

create ATTRIBUTE “Temp Scale” {
      DATATYPE H5T_C_S1
      DATA {“Celsius”}
      }
      /m1 /m2 /m3 /m4 /m5 /m6 /m7 /m8 /m9 /m10;
```

Revision History

May 27, 2010:	Version 1 draft for initial review to send to NPOESS developers
July 22, 2010:	Version 2 sent to Jonathan and Albert
July 27, 2010:	Version 3 atomicity section added. Sent to HDF5 developers
August 25, 2010:	Version 4 revision to use DDL syntax in both command line and command files. Added incremental atomcity option. Sent to HDF5 developers
September 2, 2010	Version 5 changes tool to h5edit. Sent to the public for comments.