H5edit User Guide

Version 1.1.0-beta1

1 Introduction

The h5edit tool is an HDF5 file editor. It supports commands to modify the contents of an existing HDF5 file. It enables HDF5 users to modify an HDF5 file without resorting to technical programming. Its intent is for small scale modification of the file. Data intensive changes to the file, such as those involving hundreds of data points, may not be executed in an efficient speed.

Here are some simple examples of using the H5edit tool.

```
$ h5edit -c ”DELETE /sta1/month12/temperature;” greenland.h5
```

This deletes the attribute *temperature* from dataset */sta1/month12* in the HDF5 data file *greenland.h5.*

```
$ h5edit –c ”CREATE /sta1/month12/temperature {{-40.0}};” greenland.h5
```

This creates in dataset /sta1/month12, a new attribute *temperature,* which is of the datatype of H5T_NATIVE_FLOAT and dataspace of SCALAR (single data) with the value of -40.0, in the HDF5 data file *greenland.h5.* Note that if the above two commands are executed in sequence, they have the net effect of changing the value of the attribute *temperature.* (The CHANGE command is not supported but will be implemented in the future.)

This first release supports commands for the creation and deletion of attributes of datasets and groups only. Supports for more commands and other HDF5 objects will be implemented in future releases.

2 Commands Supported

This version of H5edit supports two kinds of commands, CREATE and DELETE.

The CREATE command creates an attribute and attaches it to a specified target dataset or group in the HDF5 data file. The user needs to specify the name, datatype, dataspace, and data value of the attribute. Each command is terminated with a semicolon. Note that the CREATE example above looks so simple because it makes use of the default values of the datatype and dataspace. Here is an example that specifies the values of datatype and dataspace explicitly. Note that even the attribute name, temperature, is defined separately from its intended target-object, /sta1/month12.

```
$ h5edit --command ”CREATE /sta1/month12 temperature {DATATYPE
H5T_FLOAT_NATIVE DATASPACE SCALAR DATA {-40.0}};” greenland.h5
```

The DELETE command deletes an existing attribute from a specified dataset or group in the HDF5 data file. The user needs to specify the name of the attribute.

2.1 Dataspaces Supported

The attribute may have one of the following dataspaces. If not given, the default value is *scalar.*

2.1.1 Scalar space

This is indicated by the keyword, **SCALAR**, and it means the attribute has a single data element.

2.1.2 Dimension space

This is indicated by the keyword, **SIMPLE**, followed by the dimension sizes enclosed in a pair of parenthesis. The rank of the dimension is deduced from the number of dimension sizes specified. It means the attribute is a multiple dimensional array. For example,

SIMPLE (2,3,4)

is a 3-dimensional array of 24 data elements total.

2.1.3 Other dataspaces

The HDF5 library supports the **NULL** dataspace and also unlimited dimension sizes. Neither of them is supported by this version of H5edit.

2.2 Datatypes Supported

The attribute may have one of the following datatypes. If not given, the default value is *H5T_NATIVE_FLOAT.*

2.2.1 Integer Types

The following integer types are supported and are indicated by the corresponding keywords:

`H5T_STD_I8BE, H5T_STD_I16BE, H5T_STD_I32BE, H5T_STD_I64BE, H5T_STD_I64BE:`
Signed big endian 8, 16, 32 and 64 bits integers.

`H5T_STD_I8LE, H5T_STD_I16LE, H5T_STD_I32LE, H5T_STD_I64LE, H5T_STD_I64LE:`
Signed little endian 8, 16, 32 and 64 bits integers.

`H5T_STD_U8BE, H5T_STD_U16BE, H5T_STD_U32BE, H5T_STD_U64BE, H5T_STD_U64BE:`
Unsigned big endian 8, 16, 32 and 64 bits integers.

`H5T_STD_U8LE, H5T_STD_U16LE, H5T_STD_U32LE, H5T_STD_U64LE, H5T_STD_U64LE:`
Unsigned little endian 8, 16, 32 and 64 bits integers.

The following integer types are of the C programming language and are machine dependent. They are indicated by the following keywords:

`H5T_NATIVE_CHAR, H5T_NATIVE_UCHAR:`
Native char and unsigned char types.

`H5T_NATIVE_SHORT, H5T_NATIVE_INT, H5T_NATIVE_LONG, H5T_NATIVE_LLONG:`
Native signed short, int, long and long long type.

H5T_NATIVE_USHORT, H5T_NATIVE_UINT, H5T_NATIVE_ULONG, H5T_NATIVE_ULLONG:
Native unsigned short, int, long and long long type.

2.2.2 Floating Point Types

The following floating point types are supported and are indicated by the corresponding keywords:

H5T_IEEE_F32BE, H5T_IEEE_F64BE:
IEEE big endian 32 and 64 bits floating point types.

H5T_IEEE_F32LE, H5T_IEEE_F64LE:
IEEE little endian 32 and 64 bits floating point types.

The following floating point types are of the C programming language and are machine dependent. They are indicated by the following keywords:

H5T_NATIVE_FLOAT, H5T_NATIVE_DOUBLE, H5T_NATIVE_LDOUBLE:
Native float, double, and long double types.

2.2.3 String Types

The string type is supported and is indicated by the keyword, **H5T_STRING**. The string type consists of two more specifications, namely the size of the string and the padding mechanism.

The string size is indicated by the keyword, **STRSIZE**, followed by a positive integer value of the string size.

The padding mechanism is indicated by the keyword, **STRPAD**, following by one of the following keywords:

H5T_STR_NULLTERM:
Null terminated as in the C programming language

H5T_STR_NULLPAD:
Padded with zeros

H5T_STR_SPACEPAD:
Padded with spaces as in the Fortran programming language

Note that this version of H5edit supports only fixed size strings and null terminated padding. The other padding mechanisms will be implemented in the future.

2.2.4 Other HDF5 Types

HDF5 supports other types such as Compound, Opaque, Enum, etc. These types will be implemented in the future.

3 Command line options

3.1 Command File

When the H5edit commands are specified via the command line option, they must be specified as one single argument separated by semicolons. The command file feature will allow the commands be stored in a text file in free format. This will be easier for users as they need not worry about the shell meta-character issues. For example:

```
$ h5edit --command ”CREATE /sta1/month12/ScalarString {DATATYPE { H5T_STRING
{ STRSIZE 15 }} DATASPACE { SCALAR } DATA {\”scalar string\”}}; CREATE
/sta1/month12/ArrayString {DATATYPE { H5T_STRING { STRSIZE 10 }} DATASPACE {
SIMPLE ( 3 ) } DATA {\”an\”, \”array\”, \”string\”}}; ” greenland.h5
```

Note that the double quotes inside of the command string must be escaped so that the Unix shell will not treat them as the closing quotes. The same can be achieved by a command file which is much more readable.

```
$ h5edit --command-file strings_attributes greenland.h5
$ cat strings_attributes
CREATE /sta1/month12/ScalarString
   {DATATYPE {H5T_STRING {STRSIZE 15}}
    DATASPACE {SCALAR}
    DATA {”scalar string”}
   };
CREATE /sta1/month12/ArrayString
   {DATATYPE {H5T_STRING { STRSIZE 10}}
    DATASPACE {SIMPLE ( 3 )}
    DATA {”an”, ”array”, ”string”}
   };
```

4 Other not yet Implemented Features

The following are some other features that will be implemented in the future.

4.1 Dryrun Option

This option will allow H5edit checks the syntax correctness of the commands without making any real change to the HDF5 data file. This will be implemented in the future.

4.2 Command Atomicity

It is important to users’ production data file that the H5edit will execute the commands in an atomic manner, that is, it is either all success or no changes if there is any error. Otherwise, the HDF5 data file can be partially changed, which is not necessary desirable for all cases. Worse yet, if the H5edit fails in the middle of a command, the HDF5 file may be left in an unstable

state, resulting in a total loss of access to the remaining information in the file. This is not an acceptable behavior for production files.

The H5edit tool creates and maintains a backup copy of the original data file being edited by the tool. The Atomicity option (--atomic) controls the manner the backup copy is managed. In case of user commands errors or system failures, the data file can be recovered from the backup copy by replacing the data file with the backup copy.

When the tool starts, after it has opened the data file successfully, it will make a backup copy of the data file before applying the input commands. If the tool encounters any error, the user may recover the data file from the backup copy.

4.2.1 Backup File Name Convention

A backup copy of the data file shall have a file name that is composed of the file name of the data file but proceeded with a period and appended with ".bck". For example, if the data file name is "2010_10_01_data.h5", the backup file name will be ".2010_10_01_data.h5.bck".

4.2.2 Atomicity option

The Atomicity option, --atomic, supports 3 levels, *yes*, *no*, and *inc*. (Note that the function atomic level of *inc* is not implemented yet for this beta release. It has the same effect as the atomic level of *yes*.)

- --atomic yes applies the input commands in an all or none manner. That is, either all input commands are applied to the data file, or none is applied. This is achieved by creating the backup file at the start of the tool. If the tool encounters any error, it will exit with a non-zero code and leave the backup file behind for recovery. If the tool does not encounter any error, when it ends, it will remove the backup file and exit with a zero code.
- --atomic no will not make the backup copy and apply the input commands as much as possible. The user may use this if he is sure of the validity of the input commands or he is not concern if the data file is partially modified or corrupted.
- --atomic inc (default) will apply the changes in an all or none manner but at a command level. When a command is completed successfully, the tool will "flush" all data out to the data file and update the backup file to the same state as the data file.
  If the tool encounters any error, it will exit with a non-zero code and leave the backup file behind for recovery. If the tool does not encounter any error, when it ends, it will remove the backup file and exit with a zero code.

4.2.3 How to recover the data file in case of error

If the tool encounters some errors, it will exit with a non-zero code and leave the backup file behind. The user may first inspect the data file to see if he wishes to keep it. If he does not want to accept the modified data file, he may recover the original data file by copying the backup file over the data file.

5 Tool Command Syntax

Syntax:

`h5edit` [`-h` | `--help`]

`h5edit` *options parameter h5file*

Purpose:

An HDF5 file editor.

Description:

`h5edit` is a tool for editing an HDF5 file. The tool can read in a command file, written in the H5edit Command Language, to edit the file accordingly. Commands can be given as command line option. This is intended for simple and short commands. The H5edit Command Language is defined in "Definition of the H5edit Command Language".

Options and Parameters:

These terms are used as follows in this section:

`-h, --help`

Prints a usage message and exits.

`-c` *`command`*`, --command` *`command`*

Specifies an H5edit command to apply to the file *h5file.*

`--command-file` *`commfile`*

Specifies a command file, *`commfile`*, which contains H5edit commands written in the H5edit Command Language, to apply to the file *h5file.*

`--dryrun`

(To be supported in future implementation)

Just check the syntax of the H5edit commands against the HDF5 file without making the actual changes to the HDF5 file.

`--atomic[=`*`atomic-level`*`]`

Specifies the atomic level:

`Yes:` This is the default. It means the changes must be done as all or nothing. The original data file is restored in case of any command failures.

`no:` No atomicity is desired. Do as much changes as possible.

`inc:` This is the default. Atomicity of changes at individual command level is desired, not the entire execution.

(Note that the function atomic level of *inc* is not implemented yet for this beta release. It has the same effect as the atomic level of *yes*.)

6 Examples

6.1 Add Attributes to a File

This command adds 4 attributes to the file SVM01_ter_ grav_dev.h5. First two are unsigned short (2 bytes) attributes. The third one is a string type attribute. The last one is a floating point attribute. Note that the backslash indicates a line continuation for the Unix shell. This is needed by some Unix shells such as C shell.

```
$ h5edit -c " \

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL
{DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65528}} ; \

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL {DATATYPE
H5T_STD_U16LE DATASPACE SCALAR DATA {65535}} ; \

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurmentUnits {DATATYPE
H5T_STRING {STRSIZE 6 } DATASPACE SCALAR DATA { \"W/m^2\" }} ; \

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue {DATATYPE
H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}} ; \

" \

SVM01_ter_ grav_dev.h5
```

6.2 Delete Attributes from a File

This deletes all the attributes just created by the command above.

```
$ h5edit -c " \

DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL; \

DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL; \

DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurmentUnits; \

DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue; \" \

SVM01_ter_ grav_dev.h5
```

6.3 Using command-file option

The following 2 h5edit commands use the command-file option to do the same as the example in 6.1 and 6.2.

```
$ h5edit –command-file add_attr SVM01_ter_ grav_dev.h5

$ cat add_attr

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL
    {DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65528}};

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL
    {DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65535}};

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurmentUnits
    {DATATYPE H5T_STRING {STRSIZE 6} DATASPACE SCALAR DATA { \"W/m^2\" }};

CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue
```

```
    {DATATYPE H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}};

$ h5edit –command-file delete_attr SVM01_ter_ grav_dev.h5
$ cat delete_attr
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurmentUnits;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue;
```

As the above examples illustrate that it is more desirable to use the command-file option since one does not need to worry about meta-characters that are interpreted by the Unix Shell.

6.4 Atomicity

Say the data file, mydata.h5, contains two group attributes called /attr1 and /attr2 but not /attr_a. The following command will result in an error.

```
$ h5edit -c "DELETE /attr1; DELETE /attr_a; DELETE /attr2;” mydata.h5
```

The data file mydata.h5 is partially edited. The user may recover the data file and try again.

```
$ cp .mydata.h5.bck mydata.h5
$ rm .mydata.h5.bck
$ h5edit -c "DELETE /attr1; DELETE /attr2;” mydata.h5
```

7 Comments

Please send us your comments and suggestions to help@hdfgroup.org.

Last revised: 2013/01/03