

hdf5 package

current version 1.10

Tcl package to read and write HDF5 format files.

Description

The hdf5 package is a simple wrapper that provide some functionality of the C++ HDF5 library (www.hdfgroup.org/HDF5) at Tcl level.

HDF5 is a file format and C++ library to store datasets of numbers in an efficient way for scientific or engineering use.

It is basically a collection of *groups* and *datasets* organized in a tree shape. The *group* is similar to a directory of the filesystem.

The *dataset* is a vector of integer, floats, doubles, strings or compound data. The *dataset* can be optionally compressed. Every group or dataset is referenced in a way similar to a file in a filesystem. For example, `"/group1/groups2/dataset1"` refers to the dataset `dataset1` that is stored inside group `group2` that is stored inside group `group1`. `group1` is stored in the root of the file.

Every *group* or *dataset* can contain an arbitrary number of *attributes*, which are collections of name, value pairs.

Capabilities of the package:

It permits to read and write uni, bi, and tri-dimensional arrays of integer, float, double, string and compound types.

for multi-dimensional arrays, the size of each dimension must be specified, and is stored C-like (ordered by rows, opposite to FORTRAN-like)

For storing a set of coordinates, for example, one would store a vector of integers and three vectors of double

For efficiency the arrays are converted to and from a specialized `tcl_obj` that can hold a vector of integer, float or double. If this vector is used in TCL, it can be converted to-from a tcl string or list. When used directly from C or C++ there are not unnecessary conversions.

It is possible to create groups and list or delete objects in them. It is possible to apply or read string attributes to every dataset or group as strings. Every string is inherently stored as utf-8., but can be set as an special string HDF5 data

Commands

hdf5

hdf5 -readonly <handle> <filename>

It is the main command of this package. It Open an existent or new HDF5 filename.

Handle is an arbitrary name and becomes a new Tcl command that represents the file and that can be used to write or read from the file.

If filename does not exist, it is created. If it exists and it is already a hdf5 file, it is opened.

If not, an error is raised.

If the flag -readonly is provided, then the file is opened with read only access

<handle> set

<handle> set ?-vtype char|short|int|long|longlong|float|double|string|compound? ?-fields {<name> <subtype> <name> <subtype>...}? ?-compress 0-9? ?-dimensions {<nr> ?<nc>? ?<nd>?}? <name> <value> ?<name> <value>?

Creates or updates a dataset. name must be the full path name of the dataset (for example: /group1/group2/datasename).

value is a vector. If it is already a intarray or doublearray tcl_obj, the type is inferred from it. Otherwise, it has to be specified with option -vtype.

If -vtype is 'string' then value must be a list of strings.

-dimensions specify the number of dimensions of the array. Must be a list of one, two or three integers that specify the size of each dimension.

by default it is assumed that the number of dimensions is 1, and the size of this dimension is the amount of values

nr is the number of rows.

nd is the number of 'depths'

nc is the number of columns to represent a bidimensional array, by default is a single vector (nc==1).

The array value must contain the values ordered by rows (C-like), not by columns (FORTRAN-like), and of course the amount of data must be a multiple of nc.

The first size nr could be set to 0 to mean that this number must be according of the amount of provided values.

Instead of -dimensions {0 nc} it is possible to use (for back compatibility) the deprecated syntax -ncolumns nc (that was replaced because only allow 2 dimensional arrays).

If -vtype is 'compound' then is necessary to specify the struct components with -fields followed by a list of paired 'field name' and 'field subtype'.

'field subtype' could be: char|uchar|short|ushort|int|uint|long|ulong|llong|ullong|float|double(unsigned integers starting by 'u', and llong meaning 'long long')

<handle> get

<handle> get ?-dimensions? <name1> ?<name2> ...?

Returns a list of all named datasets. Every element of the list is either a intarray,

floatarray, doublearray or list of strings, or a list of compound types. The typical use can be:

```
lassign [handle get name1 name2] value1 value2
```

if -dimensions flag is specified then the sizes of the array on each dimension are returned instead the data.

<handle> create_group

```
<handle> create_group ?-if_not_exists? <name1> ?<name2> ...?
```

Creates one or more groups. Parent group needs to exist.

If the flag -if_not_exists is provided then a group is only created if not exists, without raise error if it already exists

<handle> delete

```
<handle> delete <name1> ?<name2> ...?
```

Deletes one or more groups or datasets.

<handle> set_attribute

```
<handle> set_attribute ?-type string|char|short|int|long|long|float|double|compound?
<obj_name> <att_name1> <value1> ?<att_name2> <value_2>...?
```

Creates one or more attributes applied to one existing group or dataset.

-type is by default set to string

<handle> get_attribute

```
<handle> get_attribute ?-type string? ?-default <def_value>? <obj_name>
?<att_name>?
```

Returns the values of one attribute. If att_name is not given, it returns a dictionary of all name value attributes that the object contains.

if option -default is given, when attribute does not exist, the default_value is returned.

Otherwise, an error is returned

-type is by default set to string

<handle> glob

```
<handle> glob ?-directory <dir>? ?-types all|group|dataset? <pattern>
```

Returns a list of groups or datasets.

Pattern can be a relative or absolute path. It can only contain substitution characters like the "*" in the last component of the path

<handle> is_group

```
<handle> is_group <name>
```

Returns true if name is a group.

<handle> is_dataset

<handle> is_dataset <name>

Returns true if name is a dataset.

<handle> is_readonly

<handle> is_readonly <name>

Returns true if name is an HDF5 file opened as read only.

<handle> close

<handle> close

Closes the file and releases the handle.

<handle> flush

<handle> flush

Forces to dump all pending buffered data in the associated file.

News

- From version 1.10
 - allow set vector attributes of types char, short, int, long, long long, float, double
(integer trick was removed, it use a vector or chars to store strings, string type must be used instead)
- From version 1.9
 - support to get/set data of more types: short, long, long long
- From version 1.8
 - new commands isreadonly and flush
 - create_group new flag -if_not_exists (e.g. create_group -if_not_exists name1 ?name2 ...?)
- From version 1.7
 - set_attribute could create compound types (e.g. set_attribute -type compound -fields {r float i float})
 - get_attribute also supporting read compound attributes
 - option -readonly to open a file for reading (default if for read and write) (e.g. hdf5 -readonly \$filename)
- From version 1.6
 - handle set replaced the flag -columns {nc} by -dimensions {nr ?nc? ?nd?}, to allow handle 3-dimensional datasets
- From version 1.5
 - get_attribute accepts the flag -type integer|string to be able to decode deprecated attributes encoded with the -integer type
- From version 1.4
 - Can create compound tables -vtype compound and -fields to define it
 - Fixed bug of is_group and is_dataset commands
 - Fixed bug of get_attribute that were returning an extra character at the end.
- From version 1.3
 - set can create multidimensional arrays, with -ncolumns
 - attributes can be written as HDF5 strings with -string
- From version 1.1
 - vtype string in command handle set, to allow the creation of string sets.
 - dimensions option in the command handle get, to know the dimensions of multidimensional arrays
- From version 1.0
 - First version.

Examples

Example:

Creation of a new file named 'test_write.h5' on the folder 'C:\temp'.

Add a new dataset named 'some_name' in the root group /.

This dataset is an unidimensional vector of a compound type, with this structure:

field name	field type
index	unsigned integer
v1	float
v2	float
v3	float

and filled with an array with two values:

```
15 3.0 2.0 1.0
```

```
2 1.1 2.2 3.3
```

Add another dataset named 'other_name', with a bidimensional array of floats, in 2 columns, and filled with:

```
15.1 3.0
```

```
22.0 1.4
```

```
6.5 2.8
```

(then the number of rows is 3)

Datasets are compressed with maximum compression level=9

Finally the file is closed

```
package require hdf5
set h hdf2
hdf5 $h {C:\temp\test_write.h5}
set compress_level 9
$h set -vtype compound -fields {index uint v1 float v2 float v3 float}
-compress $compress_level -ncolumns 1 /some_name {15 3.0 2.0 1.0 2 1.1 2.2
3.3}
$h set -vtype float -compress $compress_level -ncolumns 2 /other_name {15.1
3.0 22.0 1.4 6.5 2.8}
$h close
```

Example:

Open an existent file named 'test_read.h5' for reading.

get the attribute named /mesh/ntc1_2/all and print all its names and values

get the attribute named /mesh/ntc1_2/all and print the value of the attribute named 'type'

get the dimensions of the dataset named /mesh/ntc1_2/all/selectorOnMesh/GC011 and print them

print the data stored in this dataset

get all groups child of the root group (named \ allways), and for each group get its datasets and print its data.

```
package require hdf5
set h hdf1
hdf5 $h {C:\temo\test_read.h5}
puts [$h get_attribute /mesh/ntcl_2/all]
puts [$h get_attribute /mesh/ntcl_2/all type]
set dataset /mesh/ntcl_2/all/selectorOnMesh/GC011
puts [$h get -dimensions $dataset] ;#get the dimensions only
puts [$h get $dataset] ;#get the data
set groups [$h glob -directory \ -types group *]
foreach group $groups {
    set datasets [$h glob -directory \ $group -types dataset *]
    foreach dataset $datasets {
        set data [$h get $dataset]
        puts $data
    }
}
$h close
```