1 hdf5

package **hdf5**

current version 1.6

TCL package to read and write **hdf5** files. It permits to deal with a subset of the hdf5 options.

Description

An **hdf5** is a file format 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 *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 and bi-dimensional arrays of integer, float, double, string and compound

for bi-dimensional arrays, the number of columns 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
- The vectors 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 innecessary 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

Restrictions

In the TCL package it is only possible to deal with unidimensional vectors of integer, float or double values. Attributes can only be vectors of chars (inherently interpreted as **utf-8**) or HDF5 strings

Commands:

hdf5

```
hdf5 handle filename
```

handle becomes a 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.

handle set

handle set ?-vtype int|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 or 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 off 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 name1 ?name2 ...?

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

handle delete

handle delete name1 ?name2 ...?

Deletes one or more groups or datasets.

handle set_attribute

handle set_attribute ?-type integer|string? 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 (integer* is a deprecated option that use as trick a vector of chars ended by a NULL char, inherently interpreted as **utf-8**)

handle get_attribute

handle get_attribute ?-type integer|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 (integer* is a deprecated option that use as trick a vector of chars ended by a NULL char, inherently interpreted as **utf-8**)

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

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

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 close

handle close

Closes the file and releases the handle.

News

- 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_attibute accept 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

-vype 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

```
package require hdf5

set h hdf1 ;#the name we want for the new tcl command that will be created

#open for input and output if the file exists, else it is created for input
and output

hdf5 $h {C:\gid project
more\Proyectos\HIRF\Amelet\example_group_node_and_pointInElement
tc1_demo3.h5}

puts [$h get_attribute /mesh/ntc1_2/all] ;#return a dictionary with all
attribute names and values

puts [$h get_attribute /mesh/ntc1_2/all type] ;#return only the value of the
attribute named 'type'

set dataset /mesh/ntc1_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 \ -types dataset *]
  foreach dataset $datasets {
    set data [$h get $dataset]
    puts $data
  }
}
$h close
#creation of a new file
package require hdf5
set h hdf2
hdf5 $h {C:\gid project more\Proyectos\HIRF\Amelet\test_export_selectorOnMesh.gid\test.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}
$h close
```