H5Easy documentation

Steven Walton

May 17, 2016

Contents

1 Requirements

Make sure that you have 'libhdf5-dev' installed so that you have the correct headers. It is also suggested that you have 'h5utils' and 'hdf5-tools' installed.

2 Usage

The purpose of this library is to provide easy reading and writing to hdf5 format through the use of C++ std::vector types. It will handle most standard types but currently only writes to 'STD_U32LE', 'STD_I32LE', 'IEEE_F32LE', and 'IEEE_F64LE' (uint, int, float, and double) types. Reading will work for both big and little endian types as well as different sized uint and int data.
NOTE that casting on the fly will not work. That is, you can not request double data by calling with a float.

2.1 Compiling code

To compile run the command

```
h5c++ -std=c++11 foo.cpp -o bar
```

We use auto and thus need the c++11 standard.
Note that if you have 'Anaconda' installed that it comes with h5c++ but that it points to '/path/to/anaconda/lib' instead of the correct dev files that are installed when you install 'libhdf5-dev'. If you get undefined references check where 'h5c++' is pointing.

2.2 Write an H5 file

To do this we use a class structure. To write a vector formatted data you would do the following

```
WriteH5 data;
data.setFileName("myH5File.h5");
data.setVarName("myVariableName");
data.writeData(myVector);
```

You can also create groups

```
data.createGroup("/my/group/name");
data.setVarName("/my/group/name/myVariableName");
data.writeData(myVector);
```

With this you can also string together multiple commands.

```
WriteH5 data;
data.setFileName("myH5File.h5");
data.setVarName("var1");
data.write(vector1);
data.createGroup("/my/nested/group/");
data.setVarName("/my/nested/group/var2");
data.writeData(vector2);
data.setVarName("/my/nested/group/var3");
data.writeData(vector3);
```

On the back end we use a proxy array to store the data. This array is created on the heap to allow for bigger data to be written. This array is then deleted at the end of the getData call.

2.3 Reading H5 data

To load the data we will need to know the type that the data is stored as. This is where 'h5dump -H' becomes helpful. You can inspect the data and see the type and size.

```
LoadH5 data;
data.setFileName("myH5File.h5");
data.setVarName("myVariableName");
vector<type> loadedData = data.getData();
```

We can also handle groups similarly to above if we set the var name as the full path name. We are also able to handle 2D vector data with the following

```
vector<vector<type> > md_vec = data.getData();
```

There is more functionality written into the load data than in the write data. The Load class is able to read many different types of data. It inspects the data for the size and type. For example it will read both 'IEEE_F32LE' and 'IEEE_F32BE' data. Again we are using a proxy array to read the data that is allocated to the heap. We also use a proxy function to be able to return by the correct type. If you wish to add new types to the read then you should edit both of these sections.
Do note that we can not cast on the fly. If the data is stored as 'IEEE_F64BE' then you cannot call it with 'vector¡float¿'. As of now you must call it as a double. This is in the plans to be fixed.