RFC: h5watch

Vailin Choi

This RFC proposes a new HDF5 command line utility *h5watch.*

1 Introduction

This new command line utility *h5watch* allows users to output new records appended to a dataset as it grows. It has similar functionality like the Unix user command *tail –f* (*--follow*) option, which outputs appended data as the file grows.

2 Approach

The development of this tool will be in phases. The initial phase develops the tool's basic features for dumping appended records from one dataset. Subsequent phases will expand the dumping of records from multiple datasets and other desirable features. The input mechanism (command line syntax) as well as the output format to/from the tool will follow the HDF5 utility *h5ls*. This tool development will be done in the feature branch, *revise_chunks* as it depends on the SWMR (single write/multiple readers) feature.

For the initial phase, this new tool applies only to chunked dataset with unlimited dimension or fixed dimension with maximum dimension setting. The user cannot set H5D_ALLOC_TIME_EARLY for the dataset to be monitored; H5D_ALLOC_TIME_INCR is assumed, which is the default storage space allocation time setting. Also, while monitoring a file's dataset, user needs to ensure data being written to the dataset is flushed to the file.

3 Command line syntax and options

h5watch [OPTIONS] [OBJECT]

3.1 Options

-h [--help]	Print a usage message and exit
-V [--version]	Print version # and exit
-x [--hexdump]	Show raw data in hexadecimal format
-d [--dim]	Monitor changes in dimension size of the dataset only
–wN [--width=N]	Set the number of columns to N for output
-pN [--polling=N]	Set the polling interval to N (in seconds) when the dataset will be checked for appended data. The default polling interval is 1.

-f<list_of_fields> [--field=<list_of_fields>]

Display data for the field(s) specified in <list_of_fields> for a compound data type. The default is to display the full record. See examples in section 4.3.

<list_of_fields> can be specified as follows:

- A comma-separated list of field names in a compound data type. '.' is the default separator for a nested field.
- A single field name in a compound data type. Can use this option multiple times to select multiple fields in a compound data type.

Note that backslash is the escape character to avoid character(s) in field names that conflict with the separators used by the tool.

-s<separator> [--sep=<separator>]

Define the separator to use for inputting names in a nested field. See the '-f <list_of_fields>' option above. <separator> is a single character that separates names in the nested field. The default separator is ".". This option should precede the –f option to override the tool's default separator.

3.2 Object

[OBJECT] is the dataset to be monitored and is specified with the following format:

[<filename>/<path_to_dataset>/<dsetname>] where

<filename> is the name of the HDF5 file. It may be preceded by path separated by slashes to the specified HDF5 file.

<path_to_dataset> is the path separated by slashes to the specified dataset

<dsetname> is the name of the dataset to be monitored

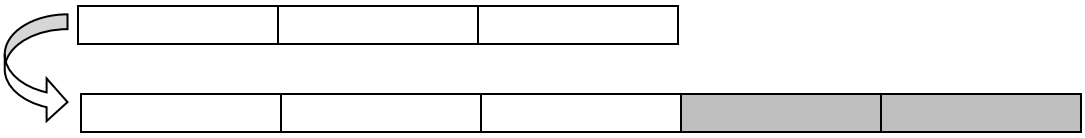
4 Examples

The dataset described in the following examples is a chunked dataset with unlimited dimensions or fixed dimensions with maximum dimension settings.

4.1 Case A: monitor one-dimensional dataset

dsetA is a one-dimensional dataset of 3 records—

- When the user changes the dimension size of *dsetA* from 3 to 5 and writes to *dsetA*:

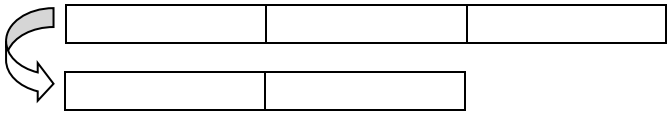


h5watch example.h5/dsetA will output the following:

```
dims[0]: 3->5

Data:
(3): record
(4): record
```

- When the user changes the dimension size of *dsetA* from 3 to 2 and writes to *dsetA*:



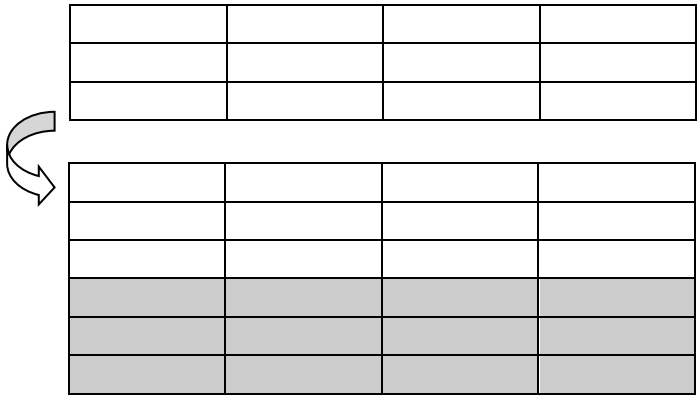
h5watch example.h5/dsetA will output the following:

```
dims[0]: 3->2
```

4.2 Case B: monitor two-dimensional dataset

dsetB is a two-dimensional dataset of 3x4 records—

- When the user changes the dimension size of *dsetB—dims[0] from* 3 to 6 and writes to *dsetB*:

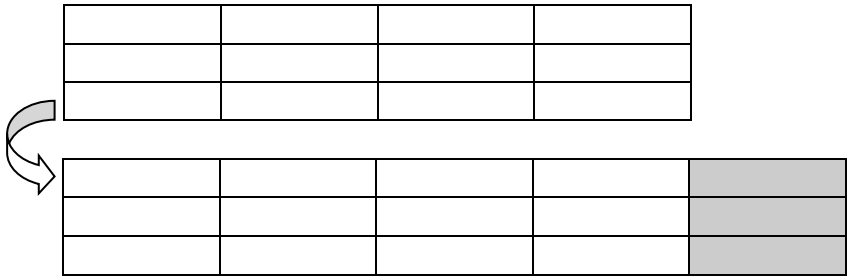


h5watch example.h5/dsetB will output the following:

```
dims[0]: 3->6
dims[1]: unchanged

Data:
(3, 0): record0, record1, record2, record3
(4, 0): record0, record1, record2, record3
(5, 0): record0, record1, record2, record3
```

- When the user changes the dimension size of *dsetB—dims[1] from* 4 to 5 and writes to *dsetB*:

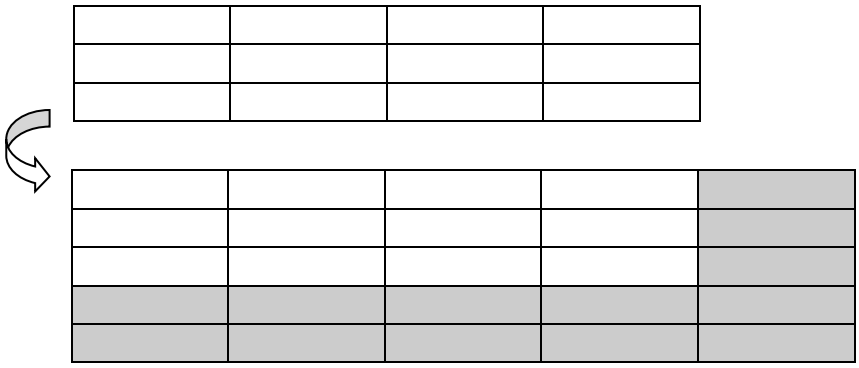


h5watch example.h5/dsetB will output the following:

```
dims[0]: unchanged
dims[1]: 4->5

Data:
(0, 4): record
(1, 4): record
(2, 4): record
```

- When the user changes the dimension size of *dsetB—dims[0] from 3 to 5, dims[1] from 4 to 5* and writes to *dsetB*:

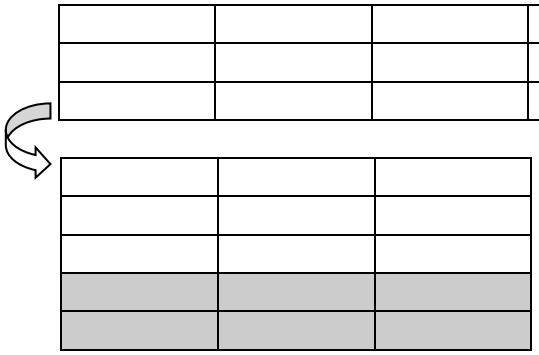


h5watch example.h5/dsetB will output the following:

```
dims[0]: 3->5
dims[1]: 4->5

Data:
(0, 4): record
(1, 4): record
(2, 4): record
(3, 0): record0, record1, record2, record3, record4
(4, 0): record0, record1, record2, record3, record4
(5, 0): record0, record1, record2, record3, record4
```

- When the user changes the dimension size of *dsetB—dims[0] from 3 to 5, dims[1] from 4 to 3* and writes to *dsetB*:

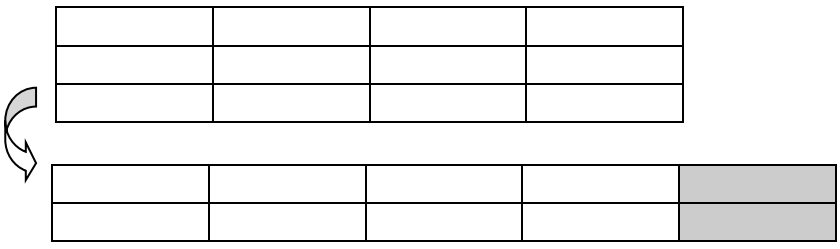


h5watch example.h5/dsetB will output the following:

```
dims[0]: 3->5
dims[1]: 4->3

Data:
(3, 0): record0, record1, record2
(4, 0): record0, record1, record2
```

- When the user changes the dimension size of *dsetB—dims[0] from 3 to 2, dims[1] from 4 to 5* and writes to *dsetB*:

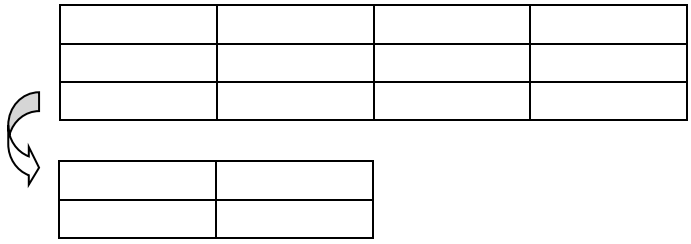


h5watch example.h5/dsetB will output the following:

```
dims[0]: 3->2
dims[1]: 4->5

Data:
(0, 4): record
(1, 4): record
```

- When the user changes the size of *dsetB—dims[0] from 3 to 2, dims[1] from 4 to 2* and writes to *dsetB*:



h5watch example.h5/dsetB will output the following:

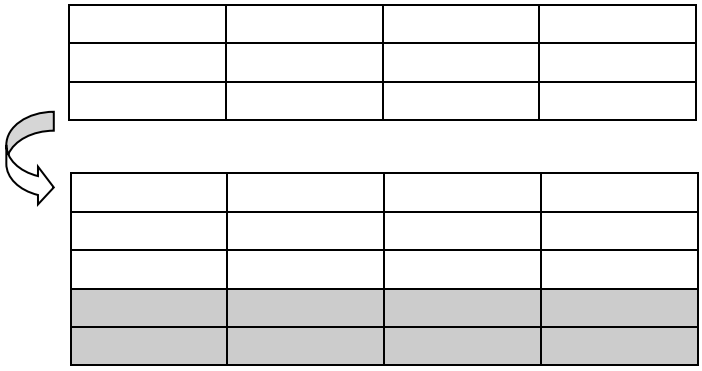
```
dims[0]: 3->2
dims[1]: 4->2
```

4.3 Case C: monitor dataset with compound data type

dsetC1 is a two-dimensional dataset of 3x4 records with compound data type defined as:

```
DATATYPE "ctype1" H5T_COMPOUND {
      H5T_STD_I32BE "c1";
      H5T_STD_I32BE "c2";
      H5T_STD_I32BE "c3";
      H5T_STD_I32BE "c4"; }
```

- When the user changes the dimension size of *dsetC1—dims[0] from 3 to 5* and writes to *dsetC1*:

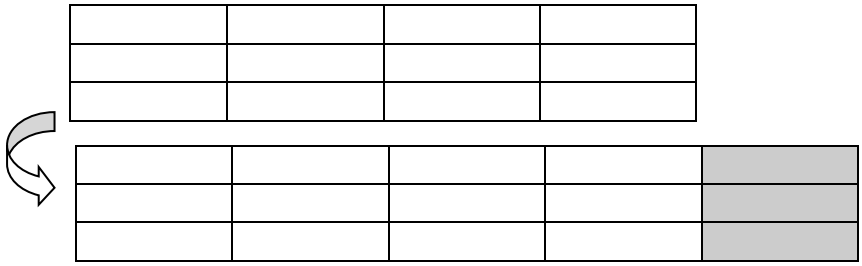


h5watch example.h5/dsetC1 will output the following:

```
dims[0]: 3->5
dims[1]: unchanged

Data:
(3, 0): {{data for c1,c2,c3,c4}, {data for c1,c2,c3,c4},
         {data for c1,c2,c3,c4}, {data for c1,c2,c3,c4}}
(4, 0): {{data for c1,c2,c3,c4}, {data for c1,c2,c3,c4},
         {data for c1,c2,c3,c4}, {data for c1,c2,c3,c4}}
```

- When the user changes the dimension size of *dsetC1-- dims[1] from 4 to 5* and writes to *dsetC1*:



h5watch –fc2,c4 example.h5/dsetC

or

h5watch –fc2 -fc4 example.h5/dsetC will output the following:

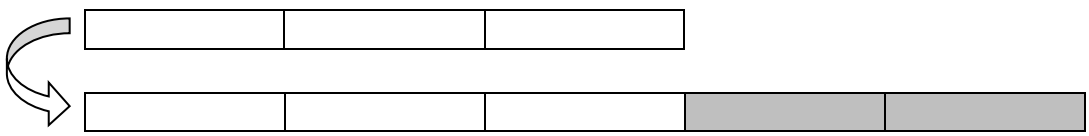
```
dims[0]: unchanged
dims[1]: 4->5

 Data:
 (0, 4): {{data for c2,c4}}
 (1, 4): {{data for c2,c4}}
 (2, 4): {{data for c2,c4}}
```

dsetC2 is a one-dimensional dataset of 3 records with nested compound data type defined as:

```
DATATYPE "ctype2" H5T_COMPOUND {
      H5T_STD_I32BE "c,1";
      H5T_STD_I32BE "c,2";
      H5_COMPOUND {
            H5T_STD_I32BE "sub.1";
            H5T_STD_I32BE "sub.2";
      } c3;
}
```

- When the user changes the dimension size of *dsetC2* from 3 to 5 and writes to *dsetC2*:



h5watch –s% -fc\,1 –fc3%sub.2 example.h5/dsetC2 will output the following:

```
dims[0]: 3->5

Data:
(3): {{data for "c,1", "sub.2"}}
(4): {{data for "c,1", "sub.2"}}
```

4.4 Case D: monitor changes in size of dataset dimensions via –d option

dsetD1 is a one-dimensional dataset of 3 records—

- When the user changes the dimension size of *dsetD1* from 3 to 5:

  h5watch –d example.h5/dsetD1 will output the following–

  dims[0]: 3→5

- When the user changes the dimension size of *dsetA* from 3 to 2:

  h5watch –d example.h5/dsetD1 will output the following–

  dims[0]: 3→2

dsetD2 is a two-dimensional dataset of 3 x 4 records—

- When the user changes the dimension size of *dsetD2—dims[0] from* 3 to 6:

  h5watch –d example.h5/dsetD2 will output the following–

  dims[0]: 3→6

  dims[1]: unchanged

- When the user changes the dimension size of *dsetD2—dims[0] from 3 to 5, dims[1] from 4 to 5*:

  h5watch –d example.h5/dsetD2 will output the following–

  dims[0]: 3→5

  dims[1]: 4→5

5 New public routines

Propose to create a new module called H5LD and add it to the HDF5 Lite API in the HDF5 High Level Interfaces. This module consists of three new public routines that perform operations associated with a dataset identifier. Similar to the existing routines in HDF5 Lite API, these routines perform more operations per call than the basic HDF5 API.

5.1 herr_t H5LDget_dset_type_size(hid_t did, char sep, char **list_of_fields, size_t *size)

This routine retrieves the size in bytes of the data type for the dataset *did* through the parameter *size*.

The parameter *list_of_fields* is an array of pointers to field names. If the first array entry is NULL, this routine just returns the dataset's data type in *size.* Otherwise, each name in the array indicates a field name in a compound data type and this routine returns the total size of the data type(s) for the selected fields. '.' is the default separator for nested compound fields and backslash is the escape character. User can choose a different separator in the parameter *sep* for nested compound fields.

The parameters for this routine are**:**

hid_t did:

IN: The dastaset identifier.

char sep:

IN: A single character that separates names in nested compound fields. If *sep* is '\0', use the default separator '.'.

*char **list_of_fields*:

IN: An array of pointers to field names for a compound data type

*size_t *size*:

OUT: The size of the dataset's data type in bytes

5.2 int H5LDget_dset_dims(hid_t did, hsize_t *cur_dims, hsize_t *max_dims)

This routine retrieves the current and maximum dataspace dimension sizes for the dataset *did* through the parameters *cur_dims* and *maxdims*. If the number of dimensions is not known, a preliminary call to this routine can be made by setting both parameters *cur_dims* and *max_dims* to NULL and the number of dimensions will be returned. Users can then allocate space for *cur_dims* and/or *max_dims* for storing the dimension sizes. This routine will retrieve dimension size(s) for non-NULL *cur_dims* and/or *max_dims* parameters.

The parameters for this routine are:

hid_t did:

IN: The dastaset identifier.

*hsize_t *cur_dims*:

OUT: The current dimension size of the dataset.

*hsize_t *max_dims:*

OUT: The maximum dimension size of the dataset.

5.3 herr_t H5LDread_dset_recs(hid_t did, hsize_t *prev_dims, hsize_t *cur_dims, char sep, char **list_of_fields, void *buf)

This routine retrieves selected data from the dataset *did* and stores the data in the parameter *buf.* The difference between the parameters *prev_dims* and *cur_dims* indicates the dimension size of the data to be selected from the dataset. Note that *cur_dims* must have at least one dimension whose size is greater than the corresponding dimension in *prev_dims*.

The parameter *list_of_fields* is an array of pointers to field names. If the first array entry is NULL, this routine returns data for the whole dataset record. Otherwise, each name in the array selects a field in a compound data type and this routine returns data for the selected fields. '.' is the default separator for nested compound fields and backslash is the escape character. User can choose a different separator in the parameter *sep* for nested compound fields.

Users can determine the size of *buf* by multiplying the size of the dataset's data type by the number of selected dataset records. The value of the parameter *size* from *H5LDget_dset_type_size()* provides the size in bytes of the dataset's data type.

The parameters for this routine are:

hid_t did:

IN: The dastaset identifier.

*hsize_t *prev_dims*:

IN: The previous dimension size of the dataset.

*hsize_t *cur_dims:*

IN: The current dimension size of the dataset.

char sep:

IN: A single character that separates names in nested compound fields. If *sep* is '\0', use the default separator '.'.

*char **list_of_fields:*

IN: An array of pointers to field names for a compound data type.

*void *buf:*

OUT: Buffer containing the selected data in the dataset.

6 Examples for the new public routines

6.1 Example 1

The dataset “DSET1” is a two-dimensional chunked dataset with atomic type defined as follows:

```
DATASET "DSET1" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 4, 13 ) / ( 60, 100 ) }
      :
      :
}
```

The following coding sample illustrates the reading of appended records to the dataset with atomic type:

```
/* open the HDF5 file */
fid = H5Fopen(FILE, H5F_ACC_RDWR, H5P_DEFAULT);

/* open the dataset */
did = H5Dopen2(fid, "DSET1", H5P_DEFAULT);

/* get the rank of the dataset */
ndims = H5LDget_dset_dims(did, NULL, NULL);
        :
        :
/* Allocate space for the arrays of dimension sizes: dims & new_dims */
        :
        :
/* get the dataset's dimension sizes */
H5LDget_dset_dims(did, dims, NULL);

/* extend the dataset by 2 for each dimension */
for (i = 0; i < ndims; i++)
      new_dims[i] = dims[i] + 2;
H5Dset_extent(did, new_dims);
        :
        :
/* Write data to the extended part of the dastaset */
        :
        :
/* define char *field_names[] = {NULL}; */
/* get the size of the dataset's data type */
H5LDget_dset_type_size(did, ‘\0’, field_names, &type_size);
        :
        :
/* Allocate buffer for storing selected records from the dataset */
/* Calculate # of selected records from dims & new_dims */
/* (type_size * number of selected records) */
        :
        :
/* read the selected records from the dataset into buf */
H5LDread_dset_recs(did, dims, new_dims, ‘\0’, field_names, buf);
        :
```

```
        :
H5Dclose(did);
H5Fclose(fid);
```

The buffer *buf* will contain data (integer values) selected from DSET1 as follows:

```
(0, 13) (0, 14) (1, 13) (1, 14) (2, 13) (2, 14) (3, 13) (3, 14)
(4, 0) (4, 1) (4, 2)……(4, 13) (4, 14)
(5, 0) (5, 1) (5, 2)……(5, 13) (5, 14)
```

6.2 Example 2

The dataset “DSET2” is a one-dimensional chunked dataset with compound type defined as follows:

```
DATASET "DSET2" {
      DATATYPE  H5T_COMPOUND {
         H5T_STD_I32LE "a";
         H5T_STD_I32LE "b";
         H5T_ARRAY { [4] H5T_STD_I32LE } "c";
         H5T_STD_I32LE "d";
         H5T_STD_I32LE "e";
         H5T_COMPOUND {
            H5T_STD_I32LE "a";
            H5T_STD_I32LE "b";
            H5T_ARRAY {[4] H5T_STD_I32LE} "c";
            H5T_STD_I32LE "d";
            H5T_STD_I32LE "e";
         } "s2";
      }
      DATASPACE  SIMPLE { ( 5 ) / ( 5 ) }
      :
      :
}
```

The following coding sample illustrates the reading of appended records to the dataset with compound data type—select only 2 fields (“d”, “s2.c”) from the compound type:

```
/* open the HDF5 file */
fid = H5Fopen(FILE, H5F_ACC_RDWR, H5P_DEFAULT);

/* open the dataset */
did = H5Dopen2(fid, "DSET2", H5P_DEFAULT);

/* get the rank of the dataset */
ndims = H5LDget_dset_dims(did, NULL, NULL);
        :
        :
/* Allocate space for the array of dimension sizes: dims & new_dims */
        :
        :
/* get the dataset's dimension sizes */
H5LDget_dset_dims(did, dims, NULL);

/* extend the dataset by 2 for each dimension */
```

```
for (i = 0; i < ndims; i++)
      new_dims[i] = dims[i] + 2;
H5Dset_extent(did, new_dims);
        :
        :
/* Write data to the extended part of the dastaset */
        :
        :
/* define char *field_names[] = {“d”, “s2%c”, NULL}; */
/* define non-default separator ‘%’ for nested field names */
/* get the size of the dataset's data type */
H5LDget_dset_type_size(did, ‘%’, field_names, &type_size);
        :
        :
/* Allocate buffer for storing selected records from the dataset */
/* Calculate # of selected records from dims & new_dims */
/* (type_size * number of selected records) */
        :
        :
/* read the selected records from the dataset into buf */
H5LDread_dset_recs(did, dims, new_dims, ‘%’, field_names, buf);
        :
        :
H5Dclose(did);
H5Fclose(fid);
```

The buffer *buf* will contain data for “d” and “s2.c” selected from DSET2 as follows:

```
record 5: {integer value for “d”, 4 integer values for array “s2.c”}
record 6: {integer value for “d”, 4 integer values for array “s2.c”}
```

7 Reference Manual

Reference manual entry for *h5watch* will be added to the HDF5 Tools Interface. Reference manual entries for the three new public routines will also be added to the HDF5 High Level Interface.

Acknowledgements

This work is supported by a commercial client of the HDF group.

Revision History

November 30, 2009: Version 1 circulated for comment within The HDF Group.