

# **Introduction to HDF5 Data Model, Programming Model and Library APIs**

**HDF and HDF-EOS Workshop VI**  
**December 4, 2002**

**December 4, 2002**

---

- **Morning session (lecture format):**
  - Introduction to HDF5
  - Programming Model
  - Introduction to Library APIs
- **Afternoon concurrent hands-on sessions**
  - Introduction to HDF5: files, groups, datasets, attributes
  - Advanced HDF5: hyperslab selections, compound datatypes, object and region references
  - Parallel HDF5
  - High-Level APIs

## Goals This Morning

---

- Introduce HDF5
- Provide a basic knowledge of how data can be organized in HDF5 & how it is used by applications.
- To provide some examples of how to read and write HDF5 files

## Goals This Afternoon

---

- Help you to start working with the HDF5 Library
  - Login NCSA machines
  - Run examples
  - Create your own programs

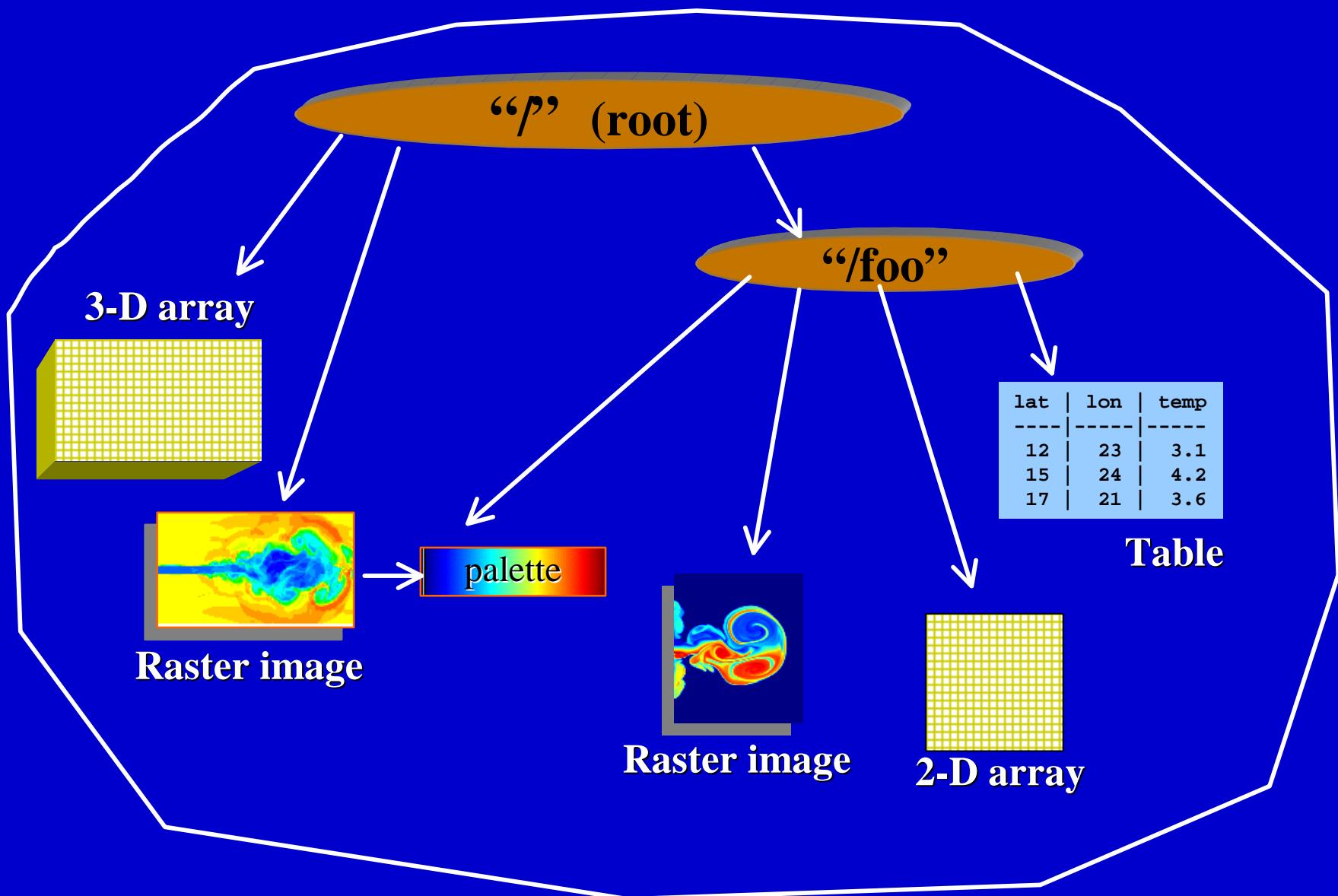
# What is HDF5?

# What is HDF5?

---

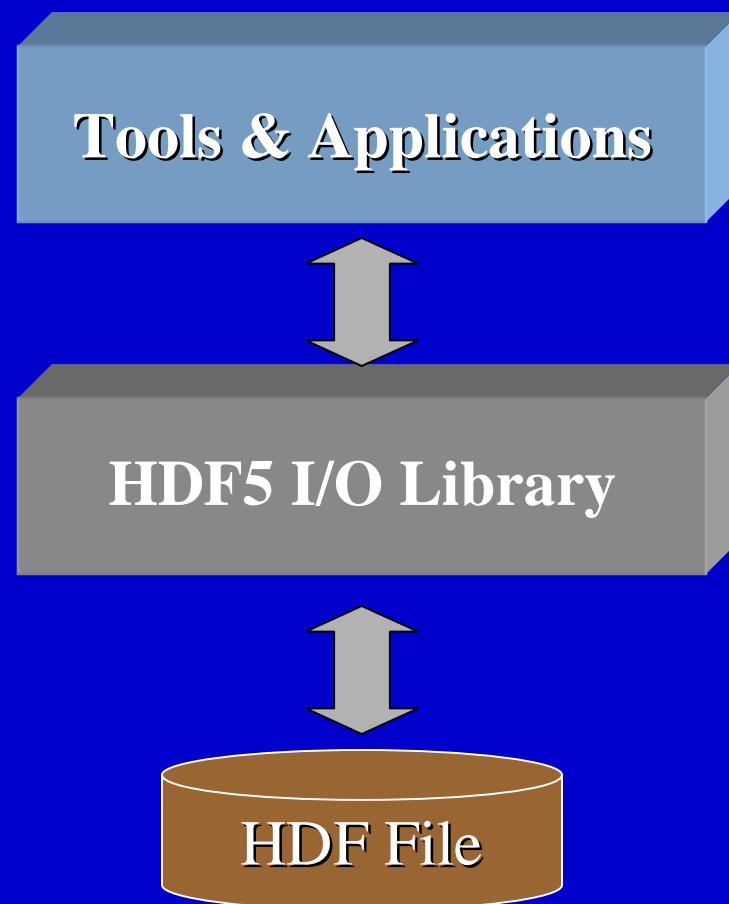
- **File format for storing scientific data**
  - To store and organize all kinds of data
  - To share data , to port files from one platform to another
  - To overcome a limit on number and size of the objects in the file
- **Software for accessing scientific data**
  - Flexible I/O library (parallel, remote, etc.)
  - Efficient storage
  - Available on almost all platforms
  - C, F90, C++ , Java APIs
  - Tools (HDFView, utilities)

# Example HDF5 file



# HDF5 Software

---



# **Overview HDF5 Data Model & I/O Library**

# HDF5 Data Model

# HDF5 file

---

- Primary Objects
  - Groups
  - Datasets
- Additional means to organize data
  - Attributes
  - Sharable objects
  - Storage and access properties

# HDF5 Dataset

---

- **Data array**
  - ordered collection of identically typed data items distinguished by their indices
- **Metadata**
  - Dataspace – rank, dimensions, other spatial info about dataset
  - Datatype
  - Attribute list – user-defined metadata
  - Special storage options – how array is organized

# Dataset Components

## Metadata

### Dataspace

Rank  
3

Dimensions  
Dim\_1 = 4  
Dim\_2 = 5  
Dim\_3 = 7

### Datatype

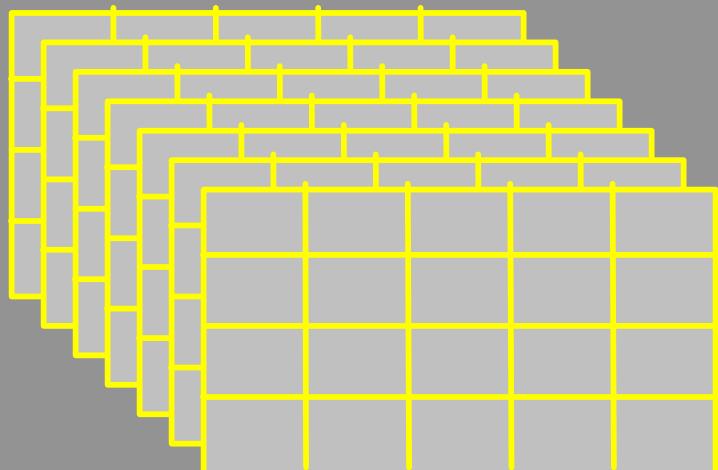
IEEE 32-bit float

### Storage info

Chunked

Compressed

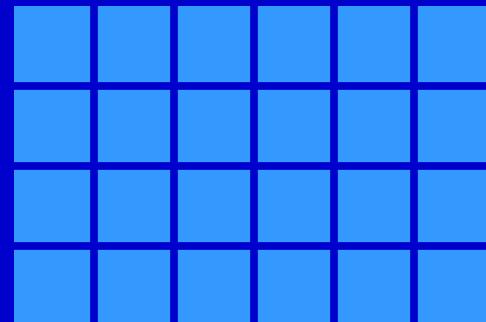
## Data



# Dataspaces

---

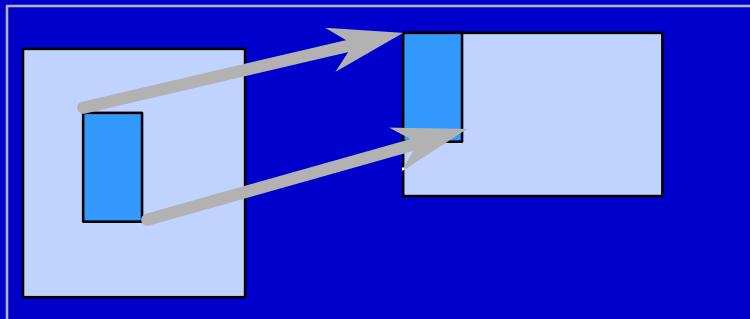
- **Dataspace – spatial info about a dataset**
  - **Rank and dimensions**
    - Permanent part of dataset definition
  - **Subset of points, for partial I/O**
    - Needed only during I/O operations
- **Apply to datasets in memory or in the file**



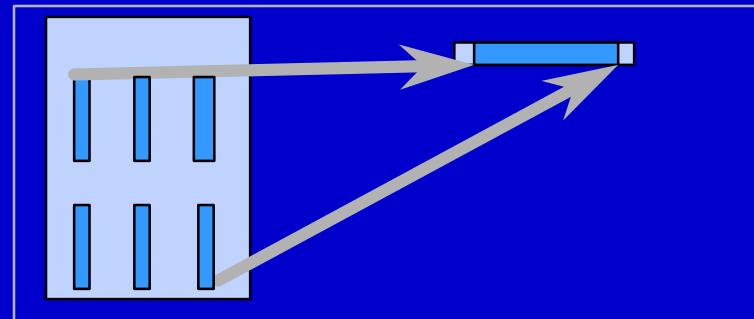
Rank = 2

Dimensions = 4x6

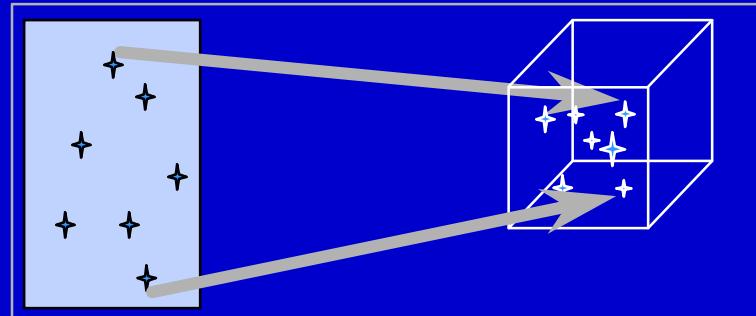
# Sample Mappings between File Dataspaces and Memory Dataspaces



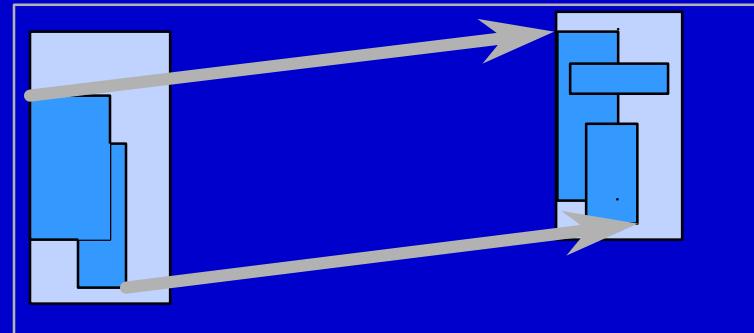
(a) Hyperslab from a 2D array to the corner of a smaller 2D array



(b) Regular series of blocks from a 2D array to a contiguous sequence at a certain offset in a 1D array



(c) A sequence of points from a 2D array to a sequence of points in a 3D array.



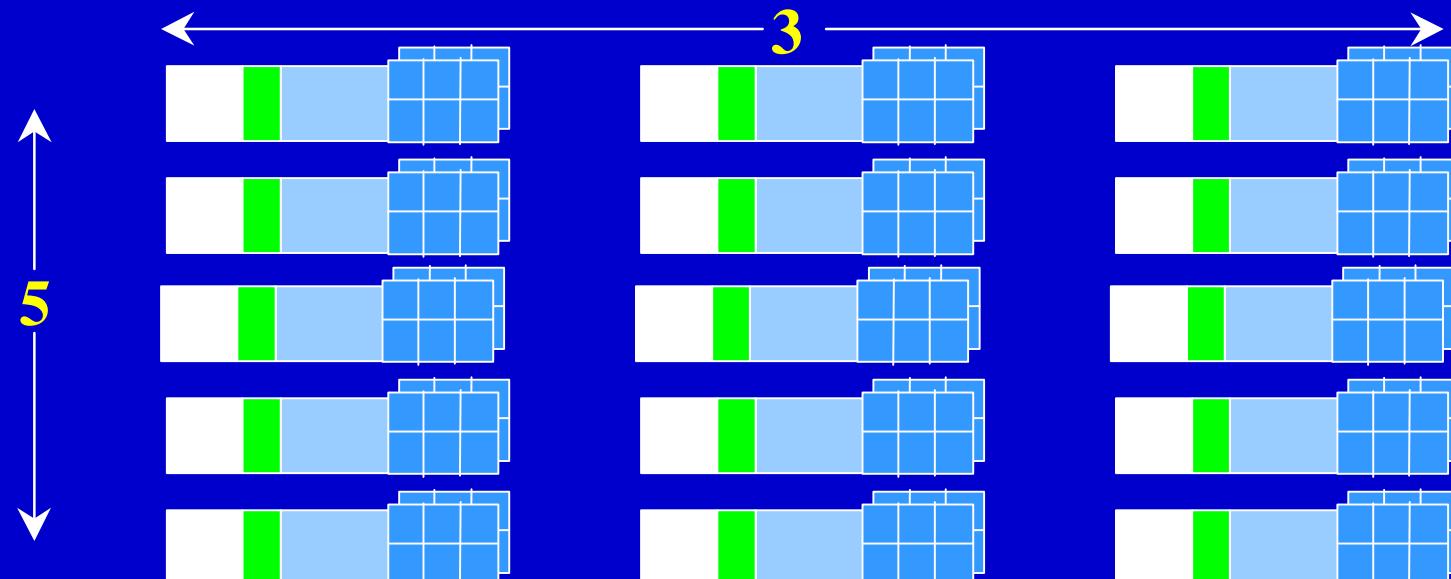
(d) Union of hyperslabs in file to union of hyperslabs in memory.

## Datatypes (array elements)

---

- **Datatype – how to interpret a data element**
  - Permanent part of the dataset definition
- **HDF5 atomic types**
  - normal integer & float
  - user-definable integer and float (e.g. 13-bit integer)
  - variable length types (e.g. strings)
  - pointers - references to objects/dataset regions
  - enumeration - names mapped to integers
  - array
- **HDF5 compound types**
  - Comparable to C structs
  - Members can be atomic or compound types

# HDF5 dataset: array of records



Datatype:

int8    int4    int16    2x3x2 array of float32

Record

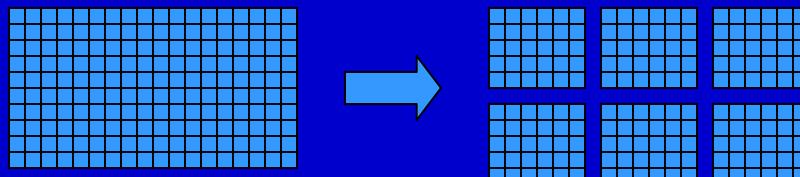
# Attributes

---

- **Attribute – data of the form “name = value”, attached to an object**
- **Operations are scaled-down versions of the dataset operations**
  - Not extendible
  - No compression
  - No partial I/O
- **Optional for the dataset definition**
- **Can be overwritten, deleted, added during the “life” of a dataset**

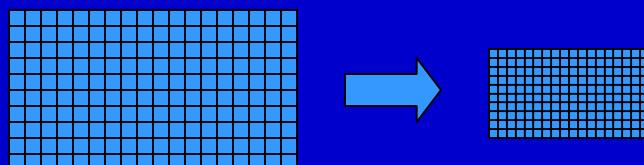
# Special Storage Options

chunked



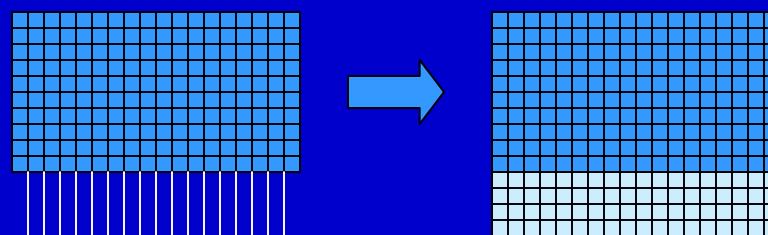
Better subsetting  
access time;  
extendable

compressed



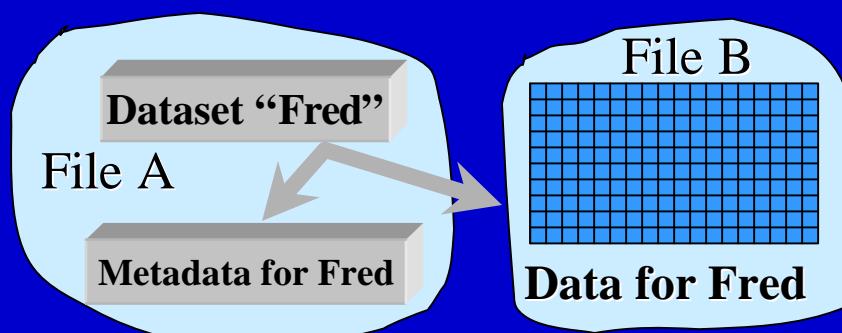
Improves storage  
efficiency,  
transmission speed

extendable



Arrays can be  
extended in any  
direction

External  
file

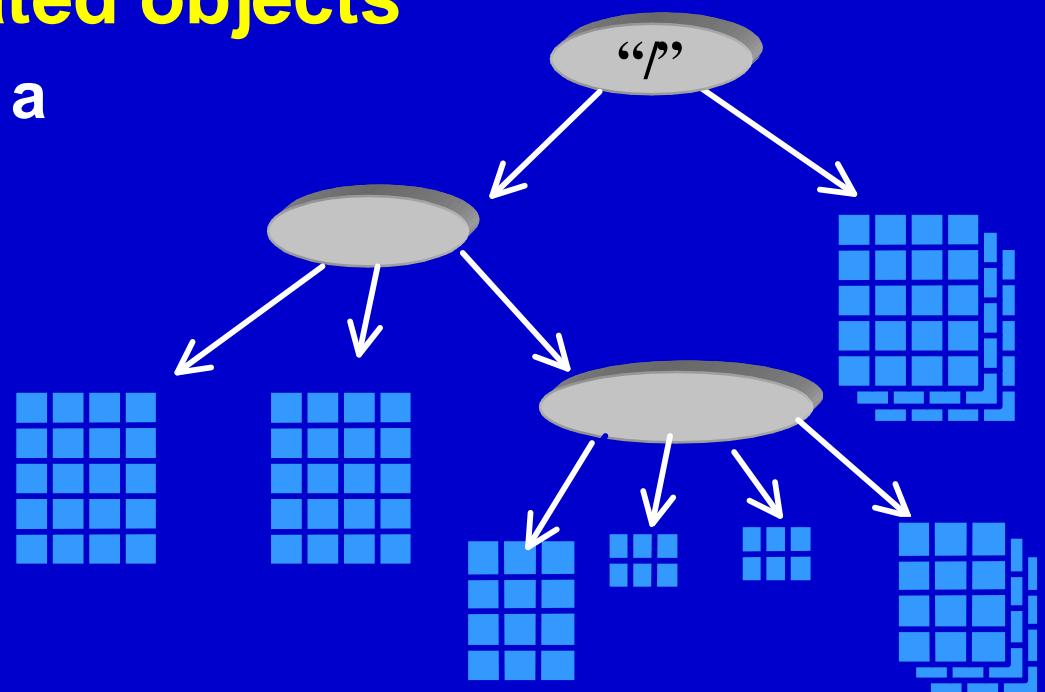


Metadata in one file,  
raw data in another.

# Groups

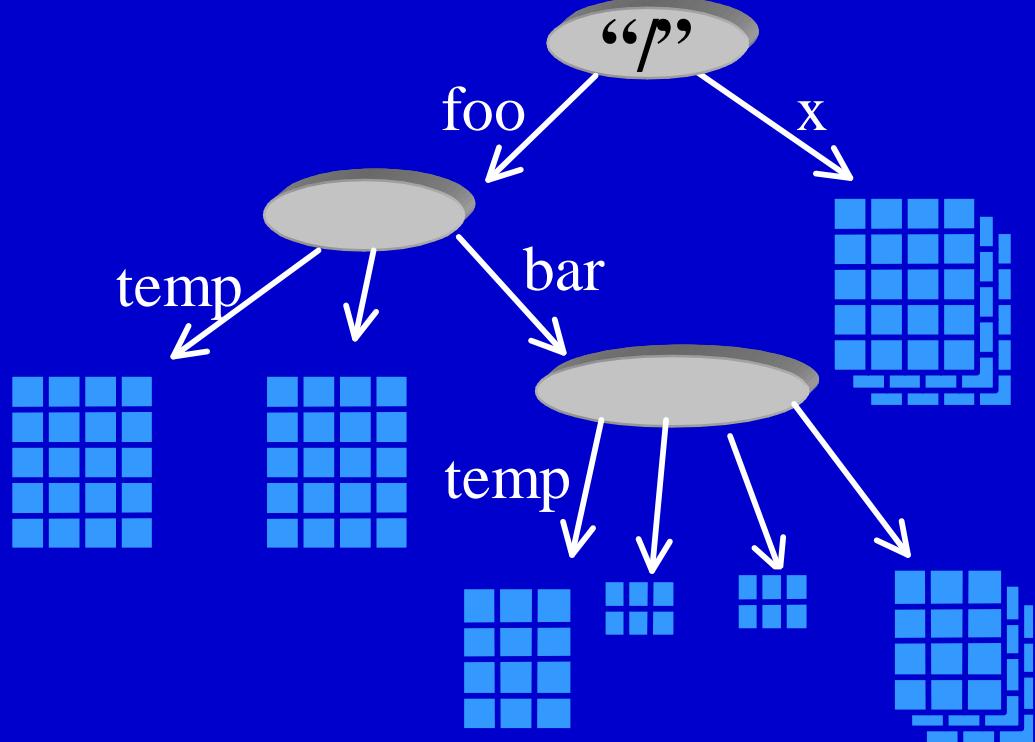
---

- **Group – a mechanism for describing collections of related objects**
- Every file starts with a root group
- Can have attributes
- Similar to UNIX directories, but cycles are allowed

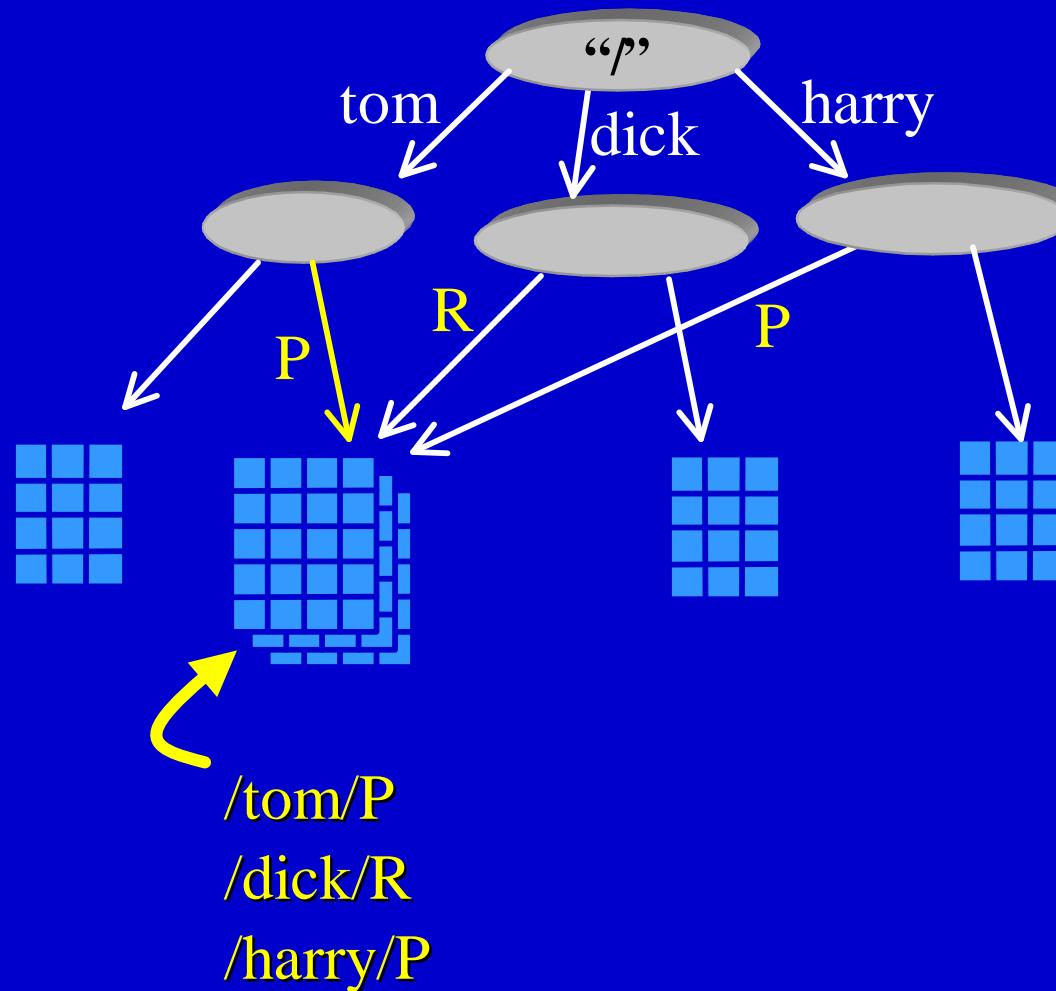


## HDF5 objects are identified and located by their pathnames

/ (root)  
/x  
/foo  
/foo/temp  
/foo/bar/temp



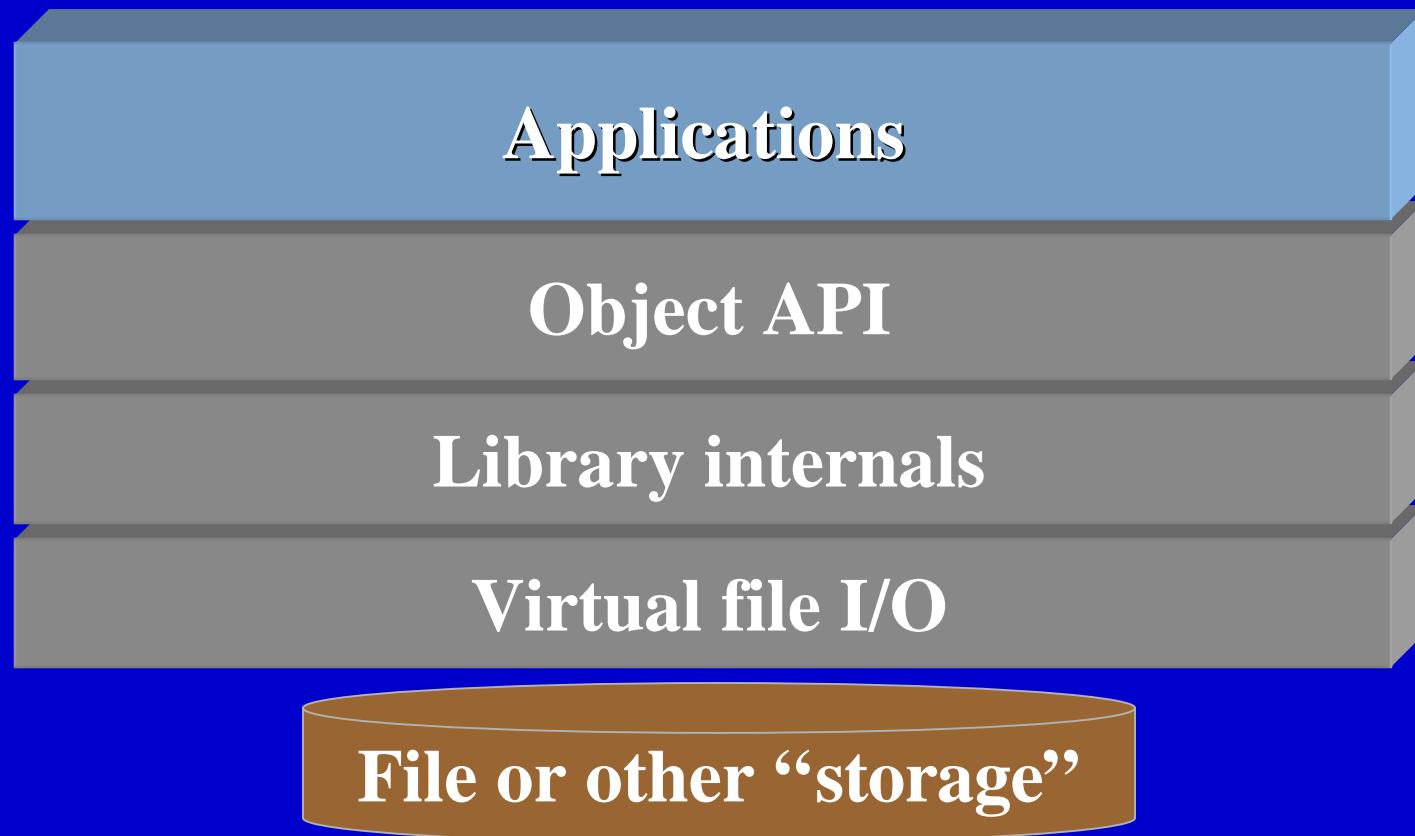
## Groups & members of groups can be shared



# HDF5 I/O Library

# Structure of HDF5 Library

---



# Structure of HDF5 Library

## Object API (C, Fortran 90, Java, C++)

- Specify objects and transformation and storage properties
- Invoke data movement operations and data transformations



## Library internals (C)

- Performs data transformations and other prep for I/O
- Configurable transformations (compression, etc.)



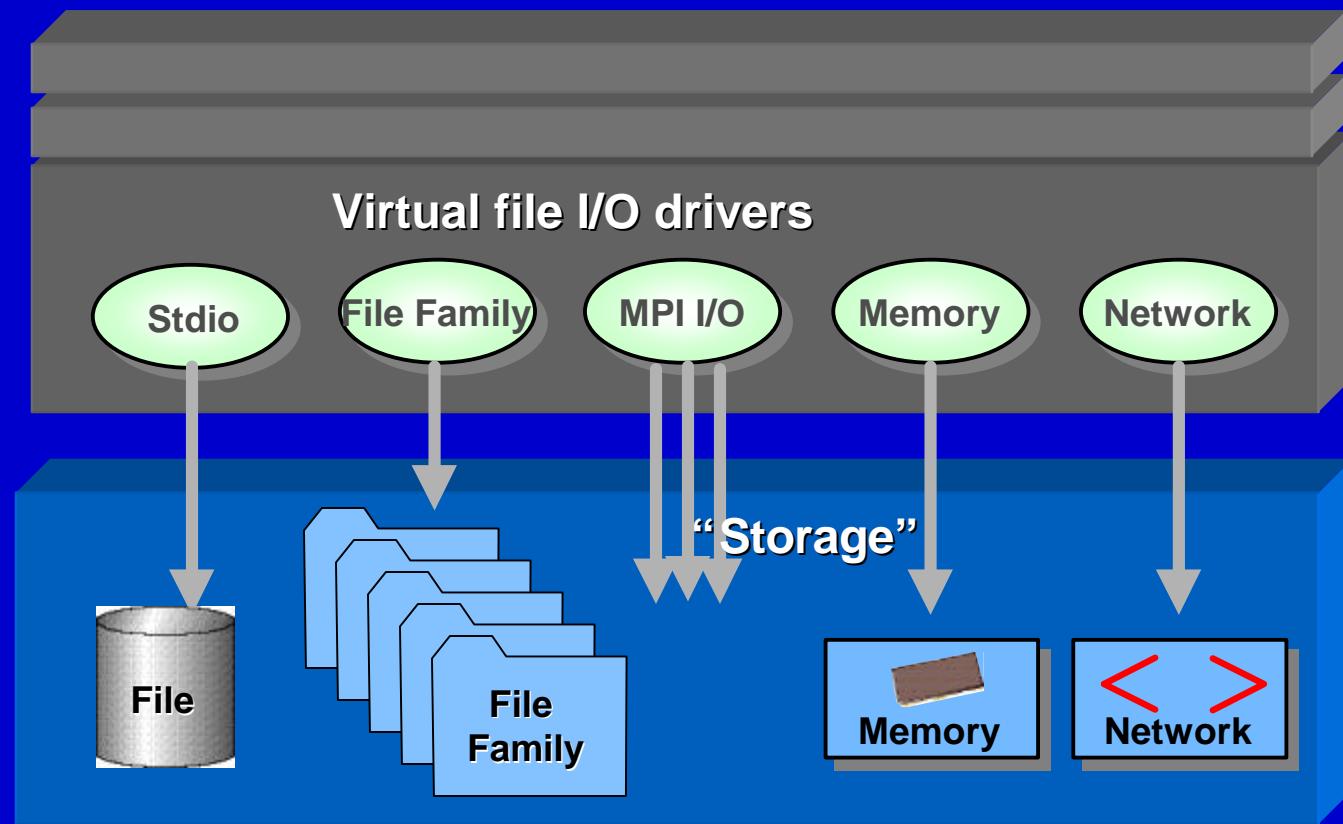
## Virtual file I/O (C only)

- Perform byte-stream I/O operations (open/close, read/write, seek)
- User-implementable I/O (stdio, network, memory, etc.)

# Virtual file I/O layer

---

- A public API for writing I/O drivers
- Allows HDF5 to interface to disk, the network, memory, or a user-defined device



# **Intro to HDF5 API**

**Programming model for sequential access**

## Goals

---

- **Describe the HDF5 programming model**
- **Give a feel for what it's like to use the general HDF5 API**
- **Review some of the key concepts of HDF5**

## General API Topics

---

- General info about HDF5 programming
- Creating an HDF5 file
- Creating a dataset
- Writing and reading a dataset

## The General HDF5 API

---

- Currently has C, Fortran 90, Java and C++ bindings.
- C routines begin with prefix H5\*, where \* is a single letter indicating the object on which the operation is to be performed.
- Full functionality

Example APIs:

H5D : Dataset interface e.g.. H5Dread

H5F : File interface e.g.. H5Fopen

H5S : dataSpace interface e.g.. H5Sclose

## The General Paradigm

---

- *Properties (called creation and access property lists) of objects are defined (optional)*
- Objects are opened or created
- Objects then accessed
- Objects finally closed

## Order of Operations

---

- The library imposes an order on the operations by argument dependencies  
*Example:* A file must be opened before a dataset because the dataset open call requires a file handle as an argument
- Objects can be closed in any order, and reusing a closed object will result in an error

## HDF5 C Programming Issues

---

For portability, HDF5 library has its own defined types:

- hid\_t:** object identifiers (*native integer*)
- hsize\_t:** size used for dimensions (*unsigned long* or *unsigned long long*)
- hssize\_t:** for specifying coordinates and sometimes for dimensions (*signed long* or *signed long long*)
- herr\_t:** function return value

For C, include `#include hdf5.h` at the top of your HDF5 application.

# **h5dump**

## **Command-line Utility for Viewing HDF5 Files**

---

**h5dump [-h] [-bb] [-header] [-a ] [-d <names>] [-g <names>]  
[-l <names>] [-t <names>] <file>**

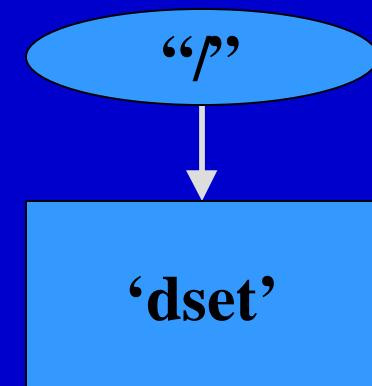
- h Print information on this command.
- header Display header only; no data is displayed.
- a <names> Display the specified attribute(s).
- d <names> Display the specified dataset(s).
- g <names> Display the specified group(s) and all the members.
- l <names> Displays the value(s) of the specified soft link(s).
- t <names> Display the specified named datatype(s).

*<names>* is one or more appropriate object names.

## Example of h5dump Output

---

```
HDF5 "dset.h5" {
GROUP "/" {
DATASET "dset" {
DATATYPE { H5T_STD_I32BE }
DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
DATA {
1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24
}
}
}
```



# **Creating an HDF5 File**

## **Steps to Create a File**

---

- **Specify File Creation and Access Property Lists, if necessary**
- **Create a file**
- **Close the file and the property lists, if necessary**

# Property Lists

---

- A **property list** is a **collection of values** that can be passed to HDF5 functions at lower layers of the library
- **File Creation Property List**
  - Controls file metadata
  - Size of the user-block, sizes of file data structures, etc.
  - Specifying H5P\_DEFAULT uses the default values
- **Access Property List**
  - Controls different methods of performing I/O on files
  - Unbuffered I/O, parallel I/O, etc.
  - Specifying H5P\_DEFAULT uses the default values.

```
hid_t H5Fcreate (const char *name, unsigned flags,  
                    hid_t create_id, hid_t access_id)
```

name	<b>IN:</b>	Name of the file to access
flags	<b>IN:</b>	File access flags
create_id	<b>IN:</b>	File creation property list identifier
access_id	<b>IN:</b>	File access property list identifier

## **herr\_t H5Fclose (hid\_t file\_id)**

file\_id      **IN:**      Identifier of the file to terminate access to

## Example 1

Create a new file using  
default properties

```
1 hid_t      file_id;
2 herr_t     status;
3 file_id = H5Fcreate ("file.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);
4 status = H5Fclose (file_id);
```

## Example 1

---

```
1 hid_t          file_id;
2 herr_t         status;
3
4 file_id = H5Fcreate ("file.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);
5
6 status = H5Fclos
```



## **h5\_crtfile.c**

```
1 #include <hdf5.h>
2 #define FILE "file.h5"
3
4 main() {
5
6     hid_t          file_id;    /* file identifier */
7     herr_t         status;
8
9     /* Create a new file using default properties. */
10    file_id = H5Fcreate (FILE, H5F_ACC_TRUNC,
11                          H5P_DEFAULT, H5P_DEFAULT);
12
13    /* Terminate access to the file. */
14    status = H5Fclose (file_id);
15 }
```

## Example 1: h5dump Output

---

```
HDF5 "file.h5" {
GROUP "/" {
}
}
```



'/'

# Create a Dataset

# Dataset Components

---

## Metadata

### Dataspace

Rank      Dimensions

3

Dim\_1 = 4

Dim\_2 = 5

Dim\_3 = 7

### Datatype

IEEE 32-bit float

### Storage info

Chunked

Compressed

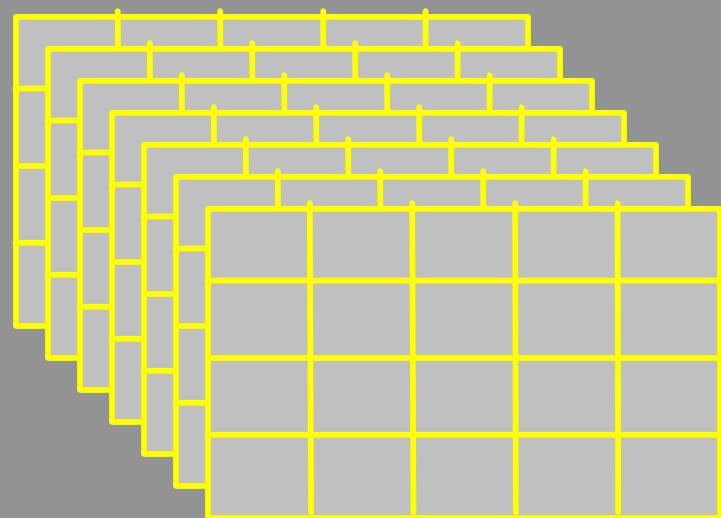
### Attributes

Time = 32.4

Pressure = 987

Temp = 56

## Data



## **Steps to Create a Dataset**

---

- Obtain location ID where dataset is to be created
- Define dataset characteristics (datatype, dataspace, dataset creation property list, if necessary)
- Create the dataset
- Close the datatype, dataspace, and property list, if necessary
- Close the dataset

## Step 1

---

**Step 1. Obtain the *location identifier* where the *dataset* is to be created**

**Location Identifier:** the file or group identifier in which to create a dataset

## Step 2

---

### **Step 2. Define the dataset characteristics**

- datatype (e.g. integer)
- dataspace (2 dimensions: 100x200)
- dataset creation properties (e.g. chunked and compressed)

# **Standard Predefined Datatypes**

---

Examples:

<b>H5T_IEEE_F64LE</b>	Eight-byte, little-endian, IEEE floating-point
<b>H5T_IEEE_F32BE</b>	Four-byte, big-endian, IEEE floating point
<b>H5T_STD_I32LE</b>	Four-byte, little-endian, signed two's complement integer
<b>H5T_STD_U16BE</b>	Two-byte, big-endian, unsigned integer

**NOTE:**

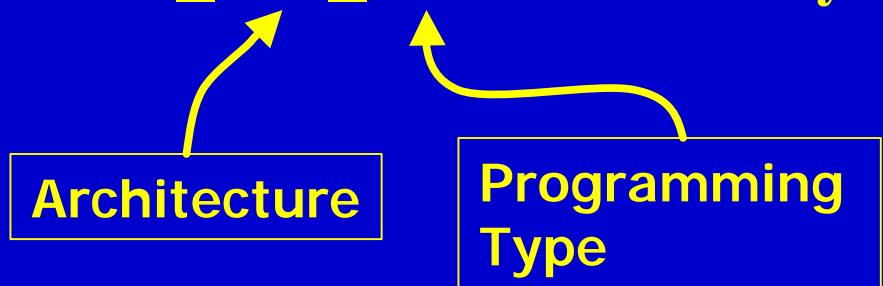
- These datatypes (DT) are the same on all platforms
- These are DT handles generated at run-time
- Used to describe DT in the HDF5 calls
- DT are not used to describe application data buffers

# Standard Predefined Datatypes

---

Examples:

- H5T\_IEEE\_F64LE** Eight-byte, little-endian, IEEE floating-point  
**H5T\_IEEE\_F32BE** Four-byte, big-endian, IEEE floating point  
**H5T\_STD\_I32LE** Four-byte, little-endian, signed two's complement integer  
**H5T\_STD\_U16BE** Two-byte, big-endian, unsigned integer



## Native Predefined Datatypes

---

Examples of predefined native types in C:

`H5T_NATIVE_INT`

(int)

`H5T_NATIVE_FLOAT`

(float )

`H5T_NATIVE_UINT`

(unsigned int)

`H5T_NATIVE_LONG`

(long )

`H5T_NATIVE_CHAR`

(char )

### NOTE:

- These datatypes are NOT the same on all platforms
- These are DT handles generated at run-time

# Dataspaces

---

- **Dataspace: size and shape of dataset and subset**
  - Dataset
    - Rank: number of dimension
    - Dimensions: sizes of all dimensions
    - Permanent – part of dataset definition
  - Subset
    - Size, shape and position of selected elements
    - Needed primarily during I/O operations
    - Not permanent
    - (Subsetting not covered in this tutorial)
- **Applies to arrays in memory or in the file**

## Creating a Simple Dataspace

```
hid_t H5Screate_simple (int rank,  
                        const hsize_t * dims,  
                        const hsize_t *maxdims)
```

rank

IN: Number of dimensions of dataspace

dims

IN: An array of the size of each dimension

maxdims

IN: An array of the maximum size of each dimension  
A value of **H5S\_UNLIMITED** specifies the  
unlimited dimension.

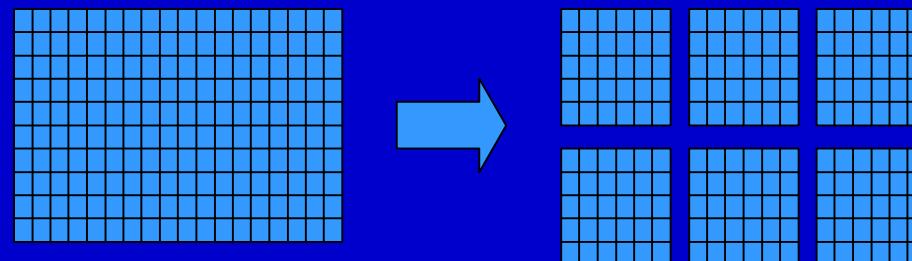
A value of **NULL** specifies that *dims* and *maxdims*  
are the same.

## Dataset Creation Property List

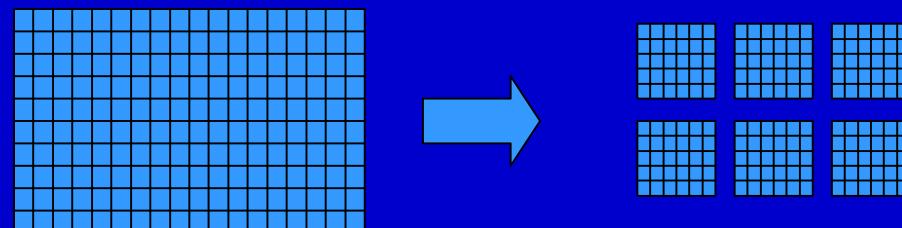
---

The *dataset creation property list* contains information on how to organize data in storage.

Chunked



Chunked & compressed



## Property List Example

---

- **Creating a dataset with ``deflate" compression**

```
create plist_id = H5Pcreate(H5P_DATASET_CREATE);  
H5Pset_chunk(create plist_id, ndims, chunk_dims);  
H5Pset_deflate(create plist_id, 9);
```

## **Remaining Steps to Create a Dataset**

---

- **Create the dataset**
- **Close the datatype, dataspace, and property list, if necessary**
- **Close the dataset**

```
hid_t H5Dcreate (hid_t loc_id, const char *name,  
                  hid_t type_id, hid_t space_id,  
                  hid_t create plist_id)
```

loc\_id

**IN:** Identifier of file or group to create the dataset  
within

name

**IN:** The name of (the link to) the dataset to create

type\_id

**IN:** Identifier of datatype to use when creating the  
dataset

space\_id

**IN:** Identifier of dataspace to use when creating  
the dataset

create plist\_id **IN:** Identifier of the dataset creation property list (or  
**H5P\_DEFAULT**)

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);

5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL);

8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                        dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Create a new file

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);
Create a dataspace

5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL);

8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                        dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);
Create a dataspace           rank           current dims
5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL)★
8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                         dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Set maxdims  
to current  
dims

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);

5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL);

8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                        dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Create a dataset

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);

5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL);
8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                         dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

Create a dataset

Dataspace

Pathname

Datatype

Property list  
(default)

## Example 2 – Create an empty 4x6 dataset

```
1 hid_t          file_id, dataset_id, dataspace_id;
2 hsize_t        dims[2];
3 herr_t         status;

4 file_id = H5Fcreate ("dset.h5", H5F_ACC_TRUNC,
                      H5P_DEFAULT, H5P_DEFAULT);

5 dims[0] = 4;
6 dims[1] = 6;
7 dataspace_id = H5Screate_simple (2, dims, NULL);

8 dataset_id = H5Dcreate(file_id,"dset",H5T_STD_I32BE,
                        dataspace_id, H5P_DEFAULT);

9 status = H5Dclose (dataset_id);
10 status = H5Sclose (dataspace_id);
11 status = H5Fclose (file_id);
```

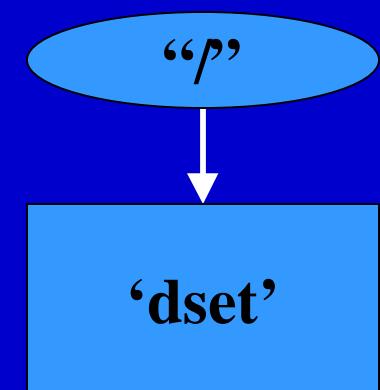
Terminate access to dataset, dataspace, & file

## Example2: h5dump Output

An empty 4x6 dataset

---

```
HDF5 "dset.h5" {
GROUP "/" {
    DATASET "dset" {
        DATATYPE { H5T_STD_I32BE }
        DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
        DATA {
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0
        }
    }
}
```



# Writing and Reading Datasets

# Dataset I/O

---

- **Dataset I/O involves**
  - reading or writing
  - all or part of a dataset
  - Compressed/uncompressed
- **During I/O operations data is translated between the source & destination (file-memory, memory-file)**
  - Datatype conversion
    - data types (e.g. 16-bit integer => 32-bit integer) of the same class
  - Dataspace conversion
    - dataspace (e.g. 10x20 2d array => 200 1d array)

# Partial I/O

---

- **Selected elements (called selections) from source are mapped (read/written) to the selected elements in destination**
- **Selection**
  - Selections in memory can differ from selection in file
  - Number of selected elements is always the same in source and destination
- **Selection can be**
  - Hyperslabs (contiguous blocks, regularly spaced blocks)
  - Points
  - Results of set operations (union, difference, etc.) on hyperslabs or points

## Reading Dataset into Memory from File

---

File

*2D array of 16-bit ints*



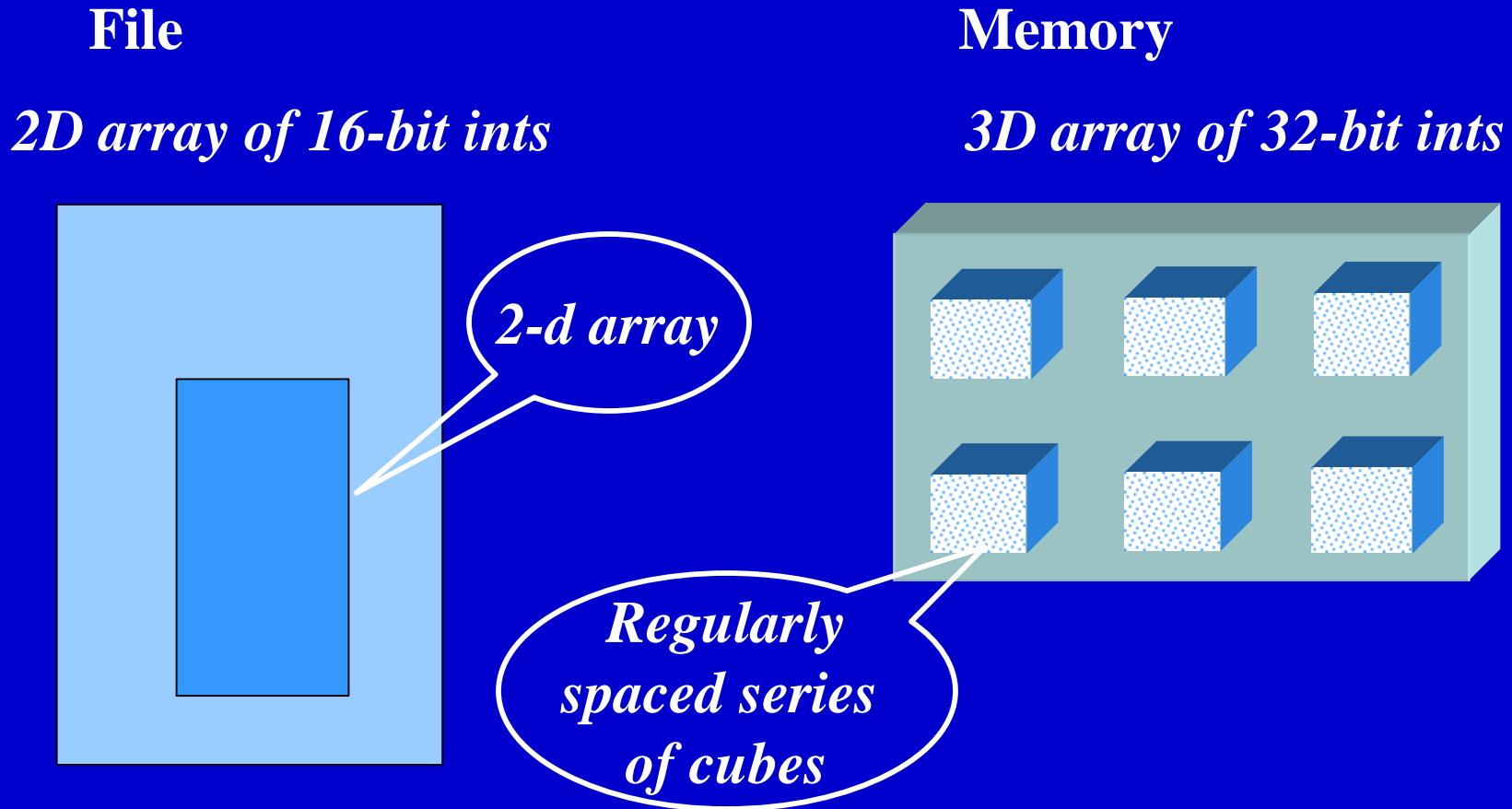
Memory

*3D array of 32-bit ints*



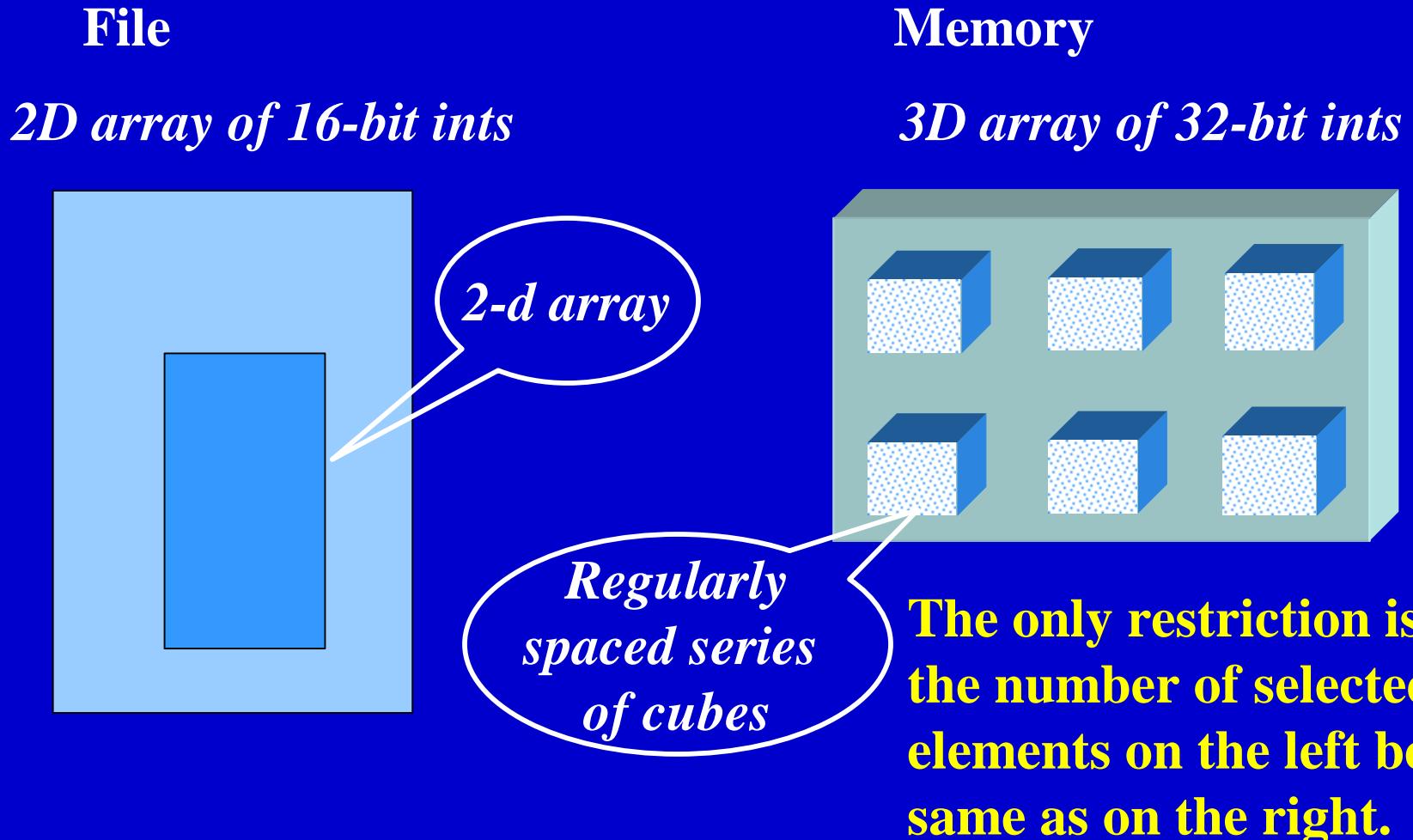
## Reading Dataset into Memory from File

---



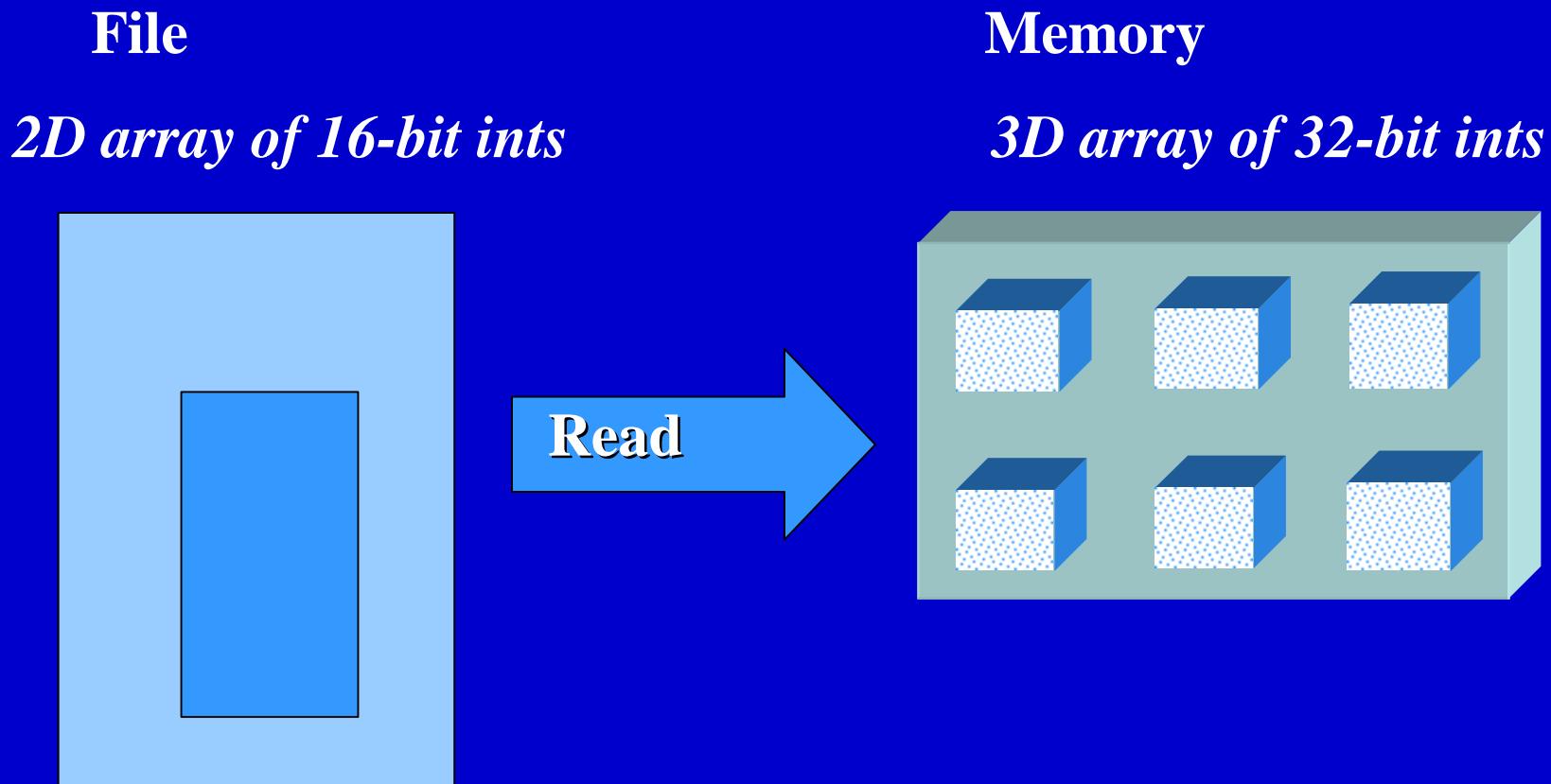
## Reading Dataset into Memory from File

---



## Reading Dataset into Memory from File

---



## Steps for Dataset Writing/Reading

---

- If necessary, open the file to obtain the file ID
- Open the dataset to obtain the dataset ID
- Specify
  - Memory datatype
  - ! *Library “knows” file datatype – do not need to specify !*
  - Memory dataspace
  - File dataspace
  - Transfer properties (optional)
- Perform the desired operation on the dataset
- Close dataspace, datatype and property lists

## Data Transfer Property List

---

The data transfer property list is used to control various aspects of the I/O, such as caching hints or collective I/O information.

## **hid\_t H5Dopen (hid\_t loc\_id, const char \*name)**

loc_id	<b>IN:</b>	Identifier of the file or group in which to open a dataset
name	<b>IN:</b>	The name of the dataset to access

**NOTE:** File datatype and dataspace are known when a dataset is opened

```
herr_t H5Dwrite (hid_t dataset_id, hid_t mem_type_id,  
                  hid_t mem_space_id, hid_t file_space_id,  
                  hid_t xfer_plist_id, const void * buf )
```

dataset\_id

**IN:** Identifier of the dataset to write to

mem\_type\_id

**IN:** Identifier of memory datatype of the dataset

mem\_space\_id

**IN:** Identifier of the memory dataspace  
(or H5S\_ALL)

file\_space\_id

**IN:** Identifier of the file dataspace (or H5S\_ALL)

xfer plist\_id

**IN:** Identifier of the data transfer properties to use  
(or H5P\_DEFAULT)

buf

**IN:** Buffer with data to be written to the file

## Example 3 – Writing to an existing dataset

```
1 hid_t          file_id, dataset_id;
2 herr_t         status;
3 int           i, j, dset_data[4][6];

4 for (i = 0; i < 4; i++)
5   for (j = 0; j < 6; j++)
6     dset_data[i][j] = i * 6 + j + 1;

7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);
8 dataset_id = H5Dopen (file_id, "dset");

9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,
                    H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

## Example 3 – Writing to an existing dataset

```
1 hid_t          file_id, dataset_id;
2 herr_t         status;
3 int           i, j, dset_data[4][6];
4
5             Initialize buffer
6
7 for (i = 0; i < 4; i++)
8     for (j = 0; j < 6; j++)
9         dset_data[i][j] = i * 6 + j + 1;
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
308
309
309
310
311
311
312
313
313
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
```

## Example 3 – Writing to an existing dataset

```
1 hid_t          file_id, dataset_id;
2 herr_t         status;
3 int           i, j, dset_data[4][6];

4 for (i = 0; i < 4; i++)
5   for (j = 0; j < 6; j++)
6     dset_data[i][j] = i * 6 + j + 1;

7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);
8 dataset_id = H5Dopen (file_id, "dset");

9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,
                    H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

Open existing file and dataset

## Example 3 – Writing to an existing dataset

```
1 hid_t          file_id, dataset_id;
2 herr_t         status;
3 int           i, j, dset_data[4][6];

4 for (i = 0; i < 4; i++)
5   for (j = 0; j < 6; j++)
6     dset_data[i][j] = i * 6 + j + 1;

7 file_id = H5Fopen ("dset.h5", H5F_ACC_RDWR, H5P_DEFAULT);
8 dataset_id = H5Dopen (file_id, "dset");

9 status = H5Dwrite (dataset_id, H5T_NATIVE_INT,
                    H5S_ALL, H5S_ALL, H5P_DEFAULT, dset_data);
```

Write to dataset

## Example 3: h5dump Output

---

```
HDF5 "dset.h5" {
GROUP "/" {
    DATASET "dset" {
        DATATYPE { H5T_STD_I32BE }
        DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
        DATA {
            1, 2, 3, 4, 5, 6,
            7, 8, 9, 10, 11, 12,
            13, 14, 15, 16, 17, 18,
            19, 20, 21, 22, 23, 24
        }
    }
}
}
```

## For more information...

---



- **HDF website**
  - <http://hdf.ncsa.uiuc.edu/>
- **HDF5 Information Center**
  - <http://hdf.ncsa.uiuc.edu/HDF5/>
- **HDF Helpdesk**
  - [hdfhelp@ncsa.uiuc.edu](mailto:hdfhelp@ncsa.uiuc.edu)
- **HDF users mailing list**
  - [hdfnews@ncsa.uiuc.edu](mailto:hdfnews@ncsa.uiuc.edu)



**Thank you**