Introduction to HDF5

Adapted from presentation by
the HDF5 Group

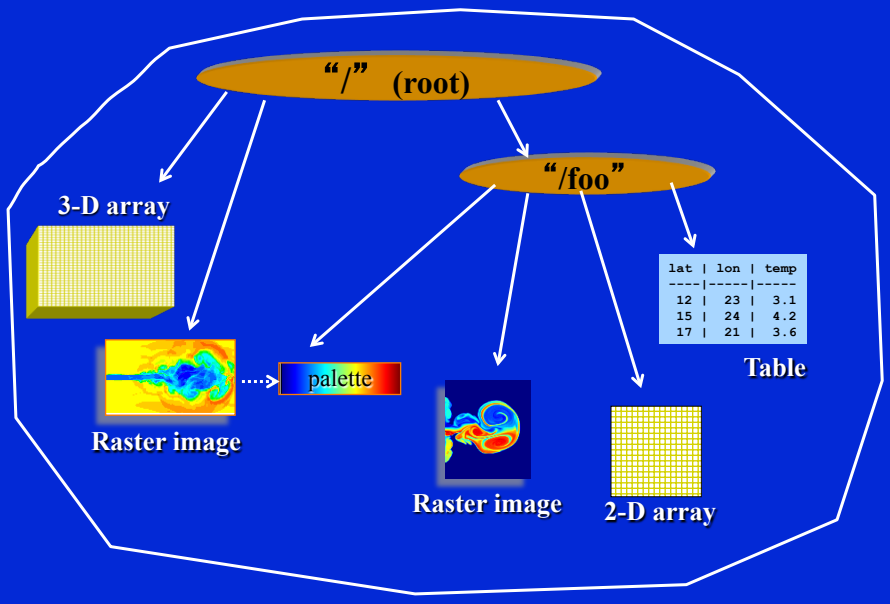
1

What is HDF5?

- File format for storing (scientific) data
  - To store and organize all kinds of (scientific) 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, Python (h5py) APIs
  - Tools (HDFView, utilities)

2

Example HDF5 file



HDF5 file

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

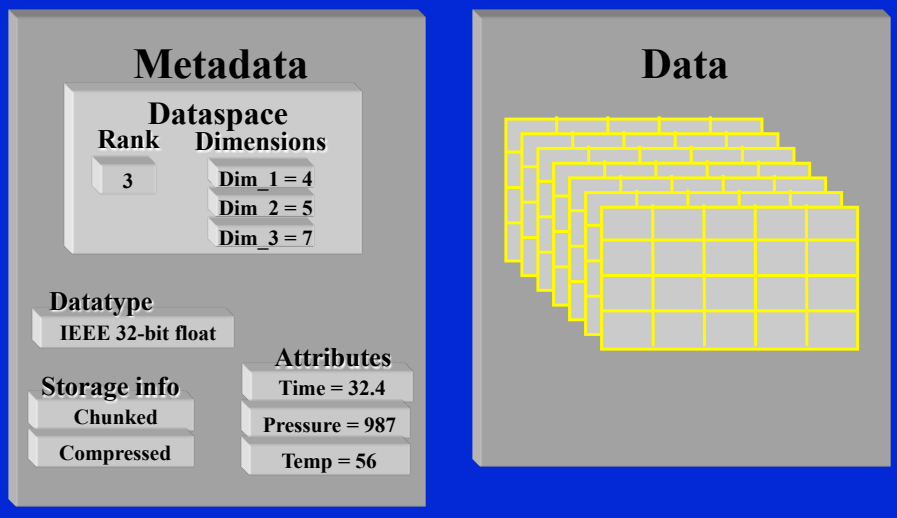
4

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

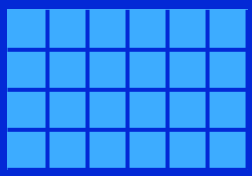
6

Dataset Components



Dataspaces

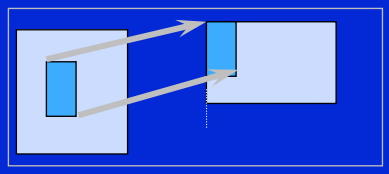
- **Dataspace – spatial info about a dataset**
  - Rank and dimensions
    - Permanent part of dataset definition
  - Subset of points, for partial I/O
    - Needed only during I/O operations
- **Apply to datasets in memory or in the file**



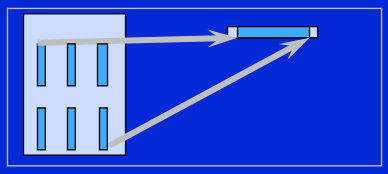
Rank = 2
Dimensions = 4x6

8

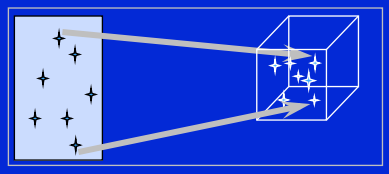
Sample Mappings between File Dataspaces and Memory Dataspaces



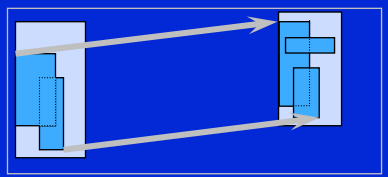
(a) Hyperslab from a 2D array to the corner of a smaller 2D array



(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.

9

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

10

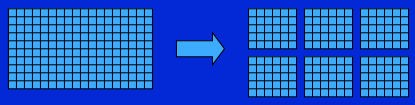
Attributes

- 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

11

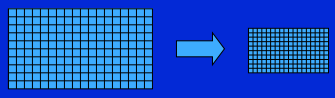
Special Storage Options

chunked



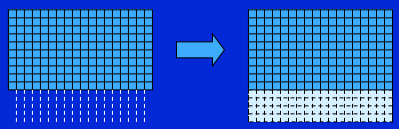
Better subsetting access time; extendable

compressed



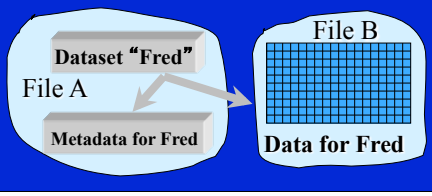
Improves storage efficiency, transmission speed

extendable



Arrays can be extended in any direction

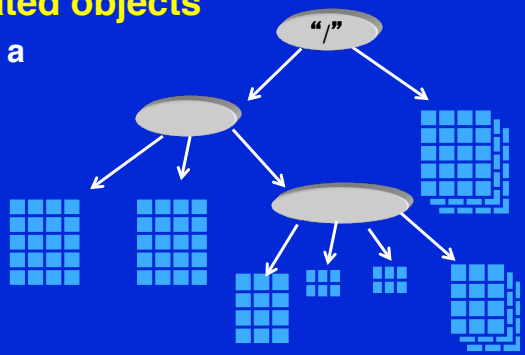
External file



Metadata in one file, raw data in another.

Groups

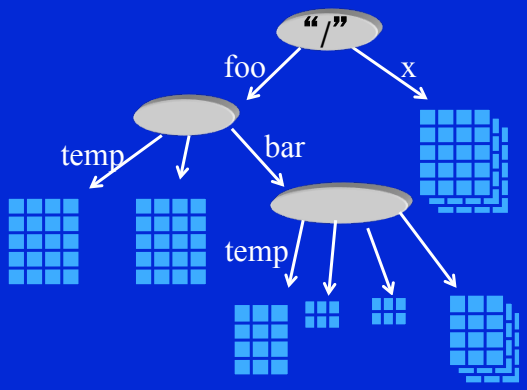
- Group – a mechanism for describing collections of related objects
- Every file starts with a root group
- Can have attributes
- Similar to UNIX directories, but cycles are allowed



13

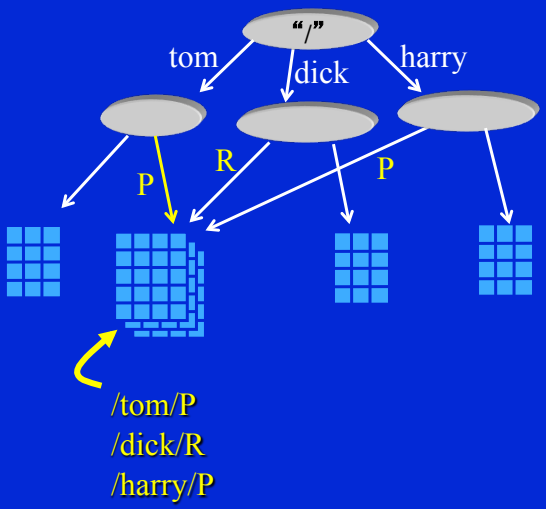
HDF5 objects and their pathnames

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



14

Shared groups & members

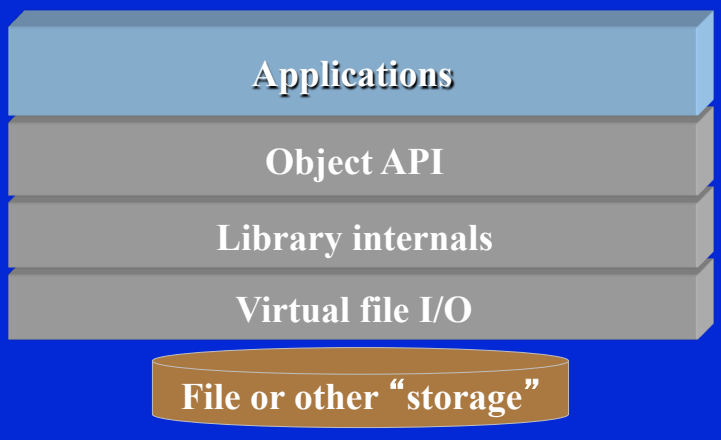


15

HDF5 I/O Library

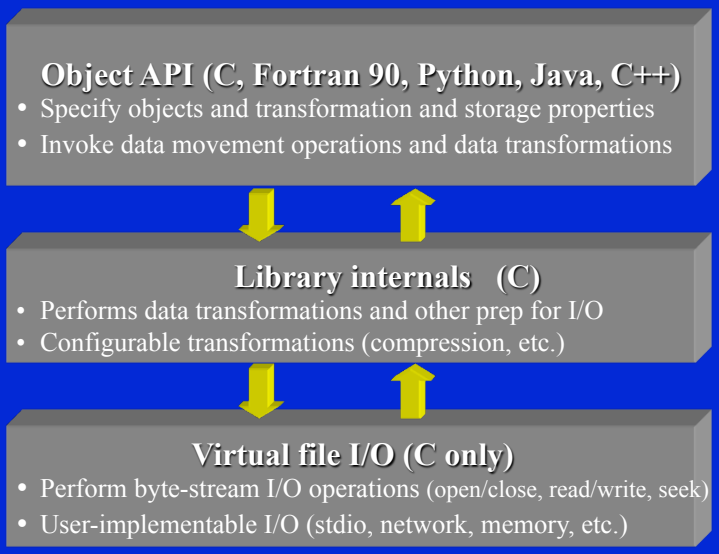
16

Structure of HDF5 Library



17

Structure of HDF5 Library



18

The General HDF5 API

- **Currently has C, Fortran 90, Python, 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**
- **Fortran: same name followed by "_f"**

Example APIs:

H5D : **D**ataset interface *e.g..* **H5Dread**
H5F : **F**ile interface *e.g..* **H5Fopen**
H5S : data**S**pace interface *e.g..* **H5Sclose**

19

The General Paradigm

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

20

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
hvl_t:	variable length datatype

For **C**, include *#include hdf5.h* at the top of your HDF5 application. For F90: USE HDF5.

21

Files

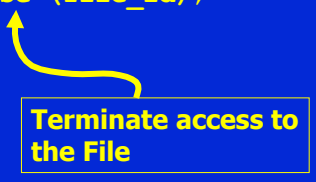
Example 1

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

4       status = H5Fclose (file_id);
```



26

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,
                             H5P_DEFAULT, H5P_DEFAULT);
11
12      /* Terminate access to the file. */
13      status = H5Fclose (file_id);
14   }
```

Example 1: h5dump Output

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

28

Groups

29

Steps to use groups

An HDF5 group is a structure containing zero or more HDF5 objects. The two primary HDF5 objects are groups and datasets. To create a group, the calling program must:

- Obtain the location identifier where the group is to be created.
- Create the group.
- Close the group.

30

hid_t H5Gcreate(*hid_t* loc_id, *const char* *name, *size_t* size_hint)

loc_id: File or parent group identifier.

name: Absolute or relative name of the new group.

hint: number of bytes to reserve for the names that will appear in the group (0 ok)

31

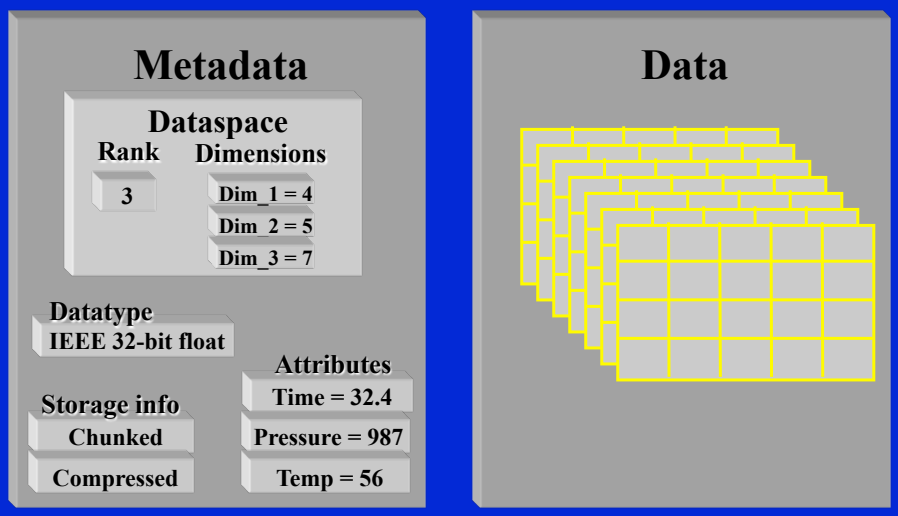
Group usage

```
CALL h5open_f(error)
!
! Create a new file using default properties.
!
CALL h5fcreate_f(filename, H5F_ACC_TRUNC_F, file_id, error)

!
! Create a group named "/MyGroup" in the file.
!
CALL h5gcreate_f(file_id, groupname, group_id, error)

!
! Close the group.
!
CALL h5gclose_f(group_id, error)

!
! Terminate access to the file.
!
CALL h5fclose_f(file_id, error)

Close FORTRAN interface.

CALL h5close_f(error)
```

32

Datasets

Dataset Components



34

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;
```

Create a new file

```
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);
```

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
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);
```

rank

current dims

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);
```

Create a dataset

```
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);

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

Create a dataset

```
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);
```

Pathname

Datatype

Dataspace

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);
```

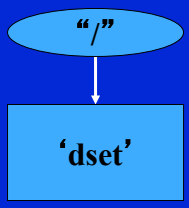
Terminate access to dataset, dataspace, & file

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

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
           }
      }
 }
 }
```



53

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)

55

Partial I/O

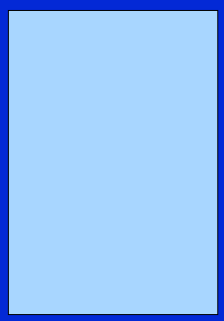
- Selected elements (called selections) from source are mapped (read/written) to the selected elements in destination
- Selection
  - Selections in memory can differ from selection in file
  - Number of selected elements is always the same in source and destination
- Selection can be
  - Hyperslabs (contiguous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

56

Reading Dataset into Memory from File

File

2D array of 16-bit ints



Memory

3D array of 32-bit ints

57

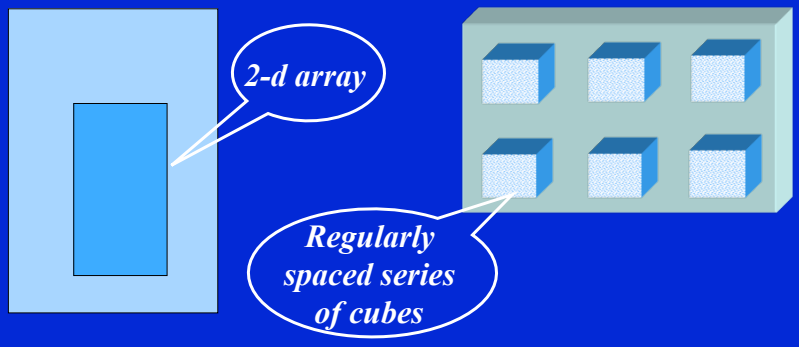
Reading Dataset into Memory from File

File

2D array of 16-bit ints

Memory

3D array of 32-bit ints



58

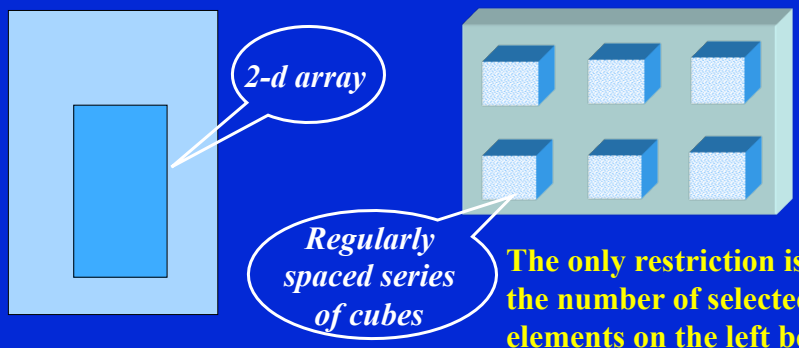
Reading Dataset into Memory from File

File

2D array of 16-bit ints

Memory

3D array of 32-bit ints



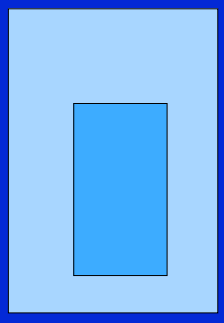
The only restriction is that the number of selected elements on the left be the same as on the right.

59

Reading Dataset into Memory from File

File

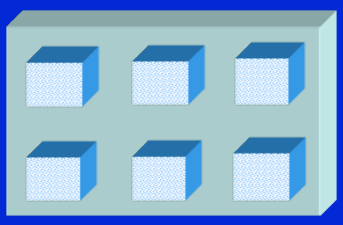
2D array of 16-bit ints



Read

Memory

3D array of 32-bit ints



60

Steps for Dataset Writing/Reading

1. If necessary, open the file to obtain the file ID
2. Open the dataset to obtain the dataset ID
3. Specify
   - Memory datatype
   - *Library "knows" file datatype – do not need to specify*
   - Memory dataspace
   - File dataspace
   - Transfer properties (optional)
4. Perform the desired operation on the dataset
5. Close dataspace, datatype and property lists

61

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];
```

Initialize buffer

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

Open existing file and dataset

```
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  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");
```

Write to dataset

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

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
      }
   }
}
}
```

69