Introduction to HDF5

Introduction to HDF5

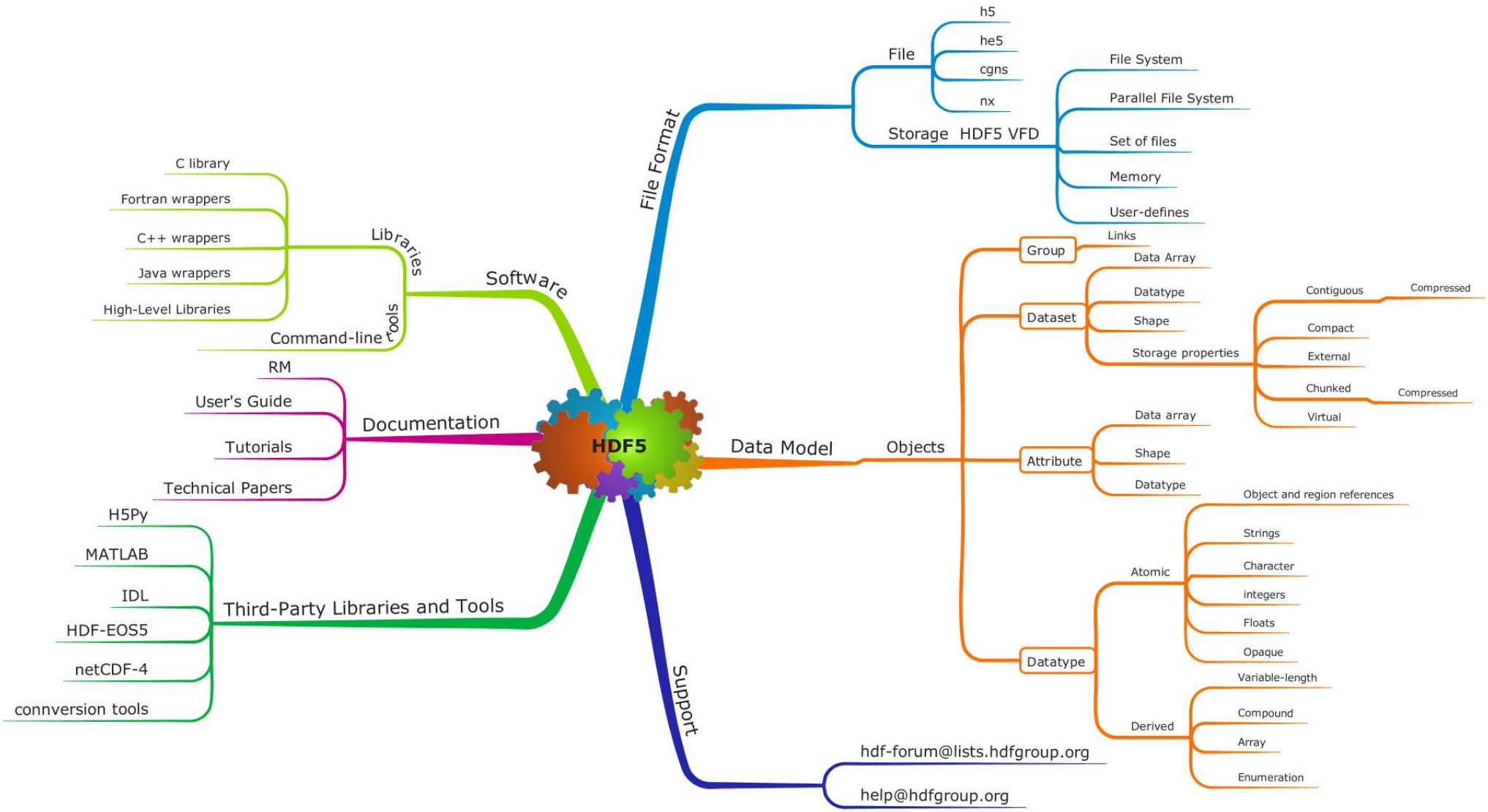
- Introduce you to HDF5
  - HDF5 data model
  - HDF5 programming model
  - Examples

WHAT IS HDF5?

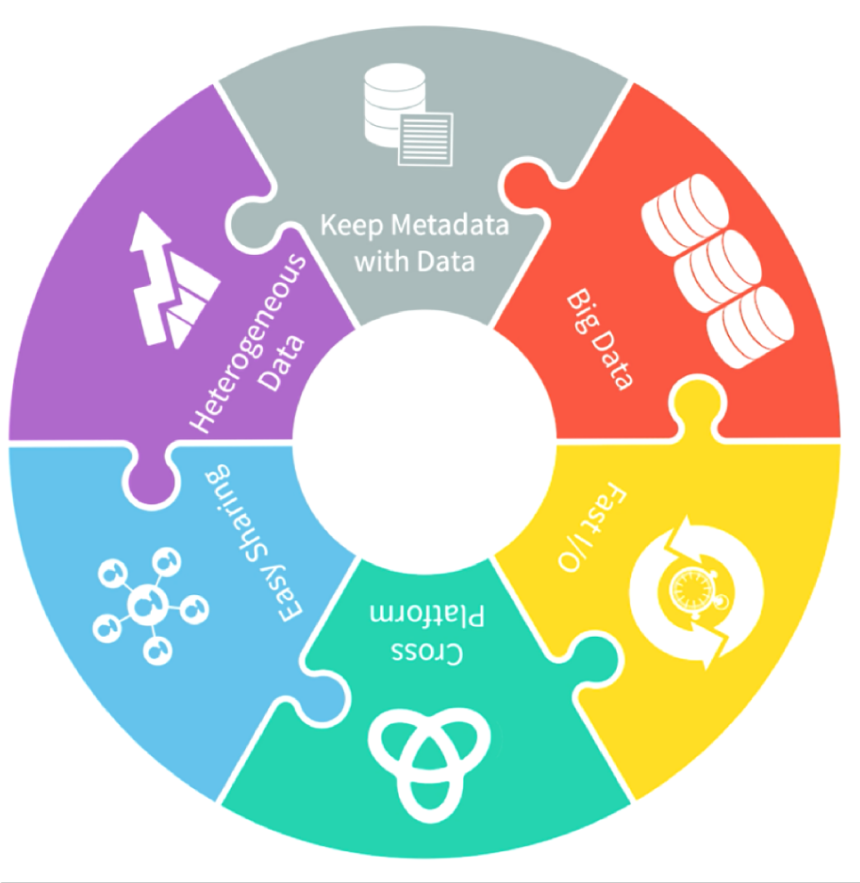
What is HDF5?

- HDF5 == Hierarchical Data Format version 5
- Open **file format**
  - Designed for high volume or complex data
- Open source **software**
  - Works with data in the format
- A **data model**
  - Structures for data organization and specification

HDF5 Mind Map



Rationale behind HDF5

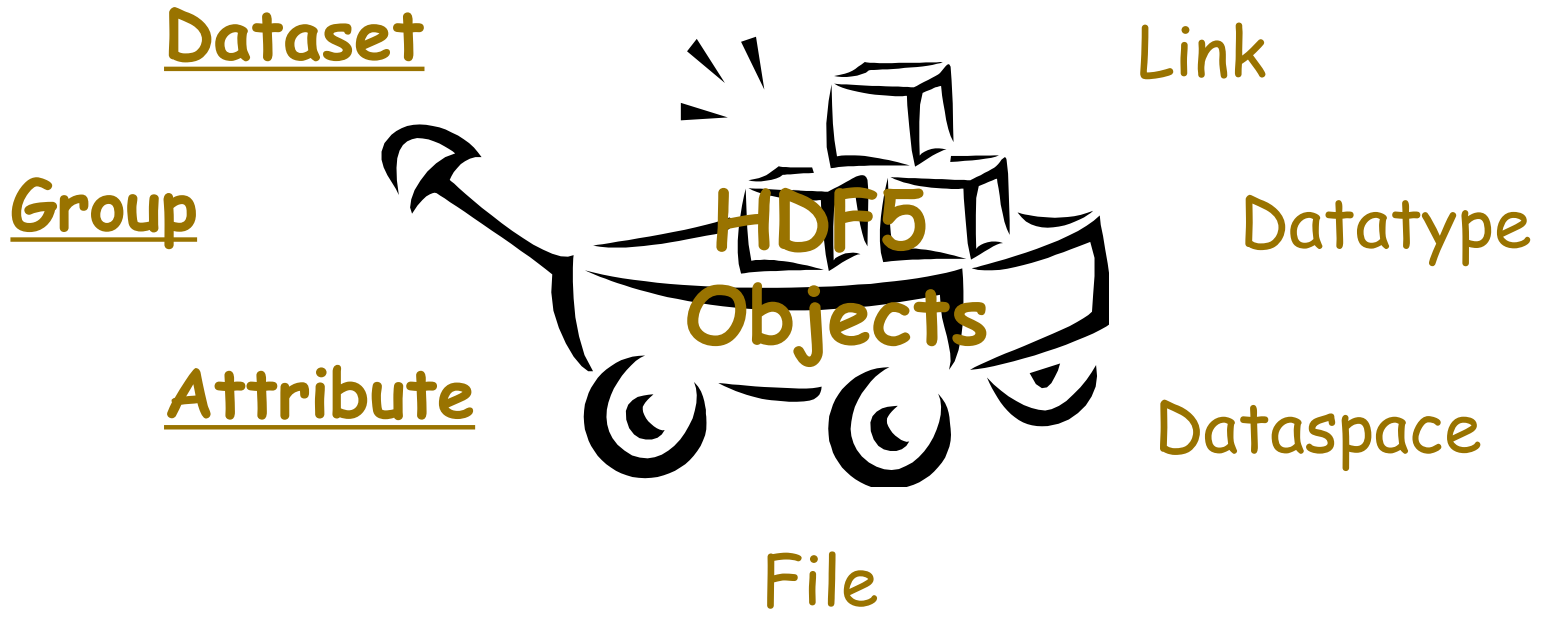


HDF5 is designed for

- Big Data - high volume and/or complex heterogeneous data
- Every size and type of system (portable, shareable)
- Flexible, efficient storage and I/O
- Enabling applications to evolve in their use of HDF5 and to accommodate new models
- Supporting long-term data preservation

HDF5 DATA MODEL

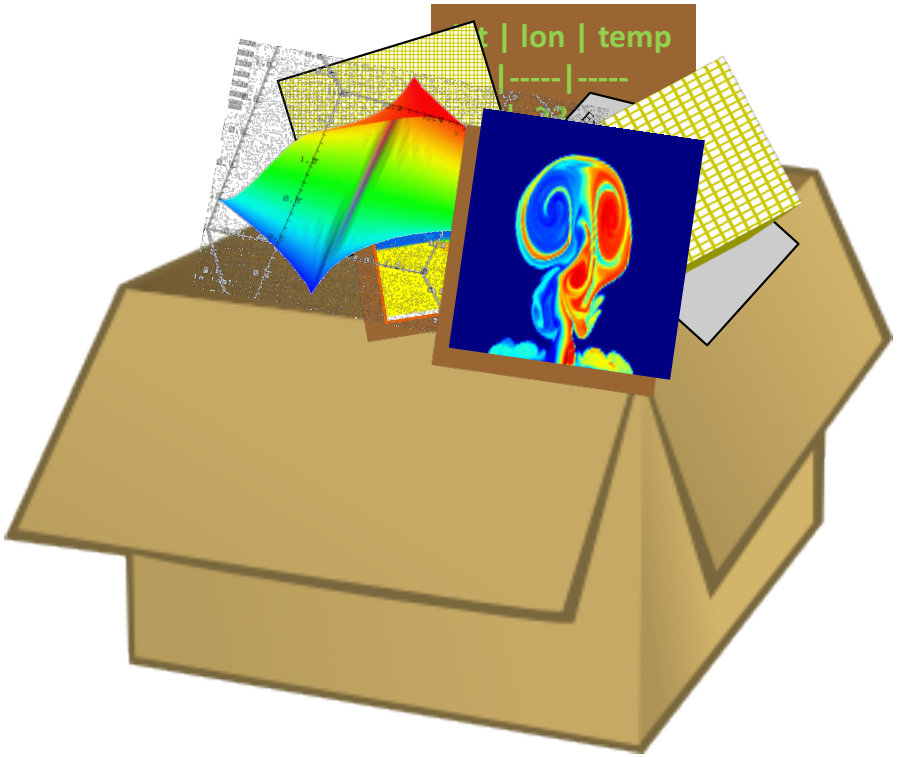
HDF5 Data Model



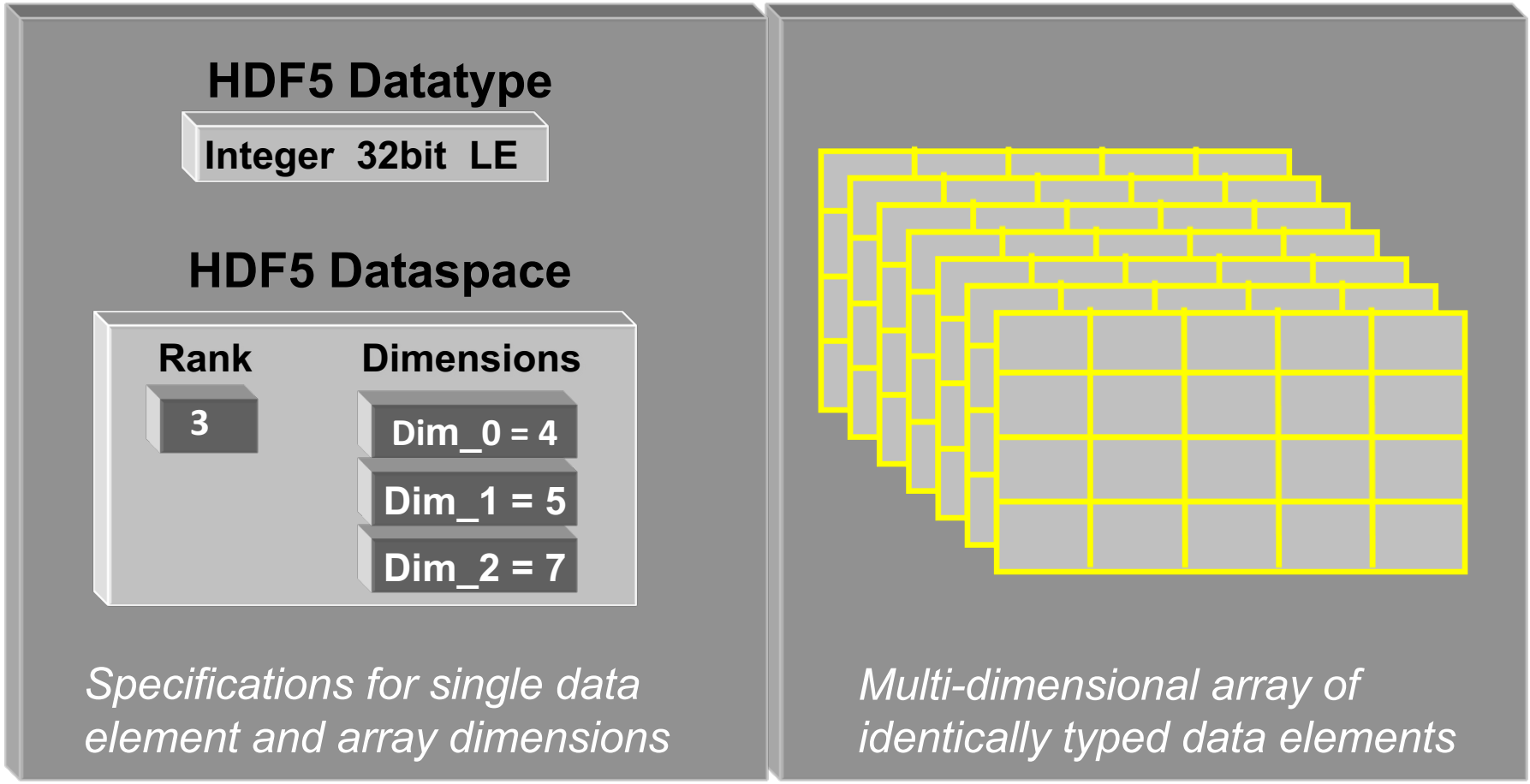
a.k.a. HDF5 Abstract Data Model
a.k.a. HDF5 Logical Data Model

HDF5 File

An HDF5 file is a **container** that holds data objects.



HDF5 Dataset



- HDF5 datasets **organize and contain** "raw data values."
  - HDF5 datatype describes individual data elements.
  - HDF5 dataspace describes the logical layout of the data elements.

HDF5 Dataspace

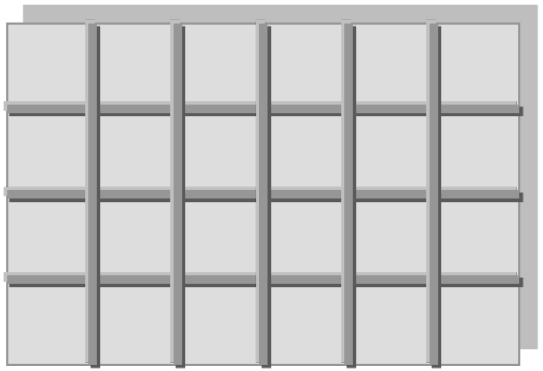
- Describes the logical layout of the elements in an HDF5 dataset
  - NULL
    - no elements
  - Scalar
    - single element
  - Simple array (*most common*)
    - multiple elements organized in a rectangular array
      - rank = number of dimensions
      - dimension sizes = number of elements in each dimension
      - maximum number of elements in each dimension
        - may be fixed or unlimited

HDF5 Dataspace

Two roles:

Dataspace contains spatial information

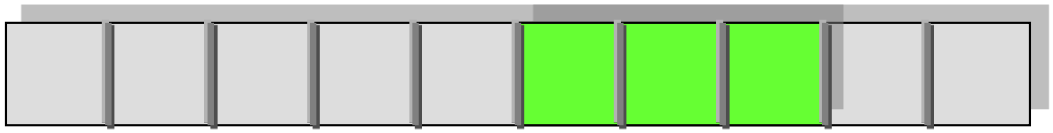
- Rank and dimensions
- Permanent part of dataset definition



Rank = 2

Dimensions = 4x6

Partial I/0: Dataspace describes application's data buffer and data elements participating in I/O



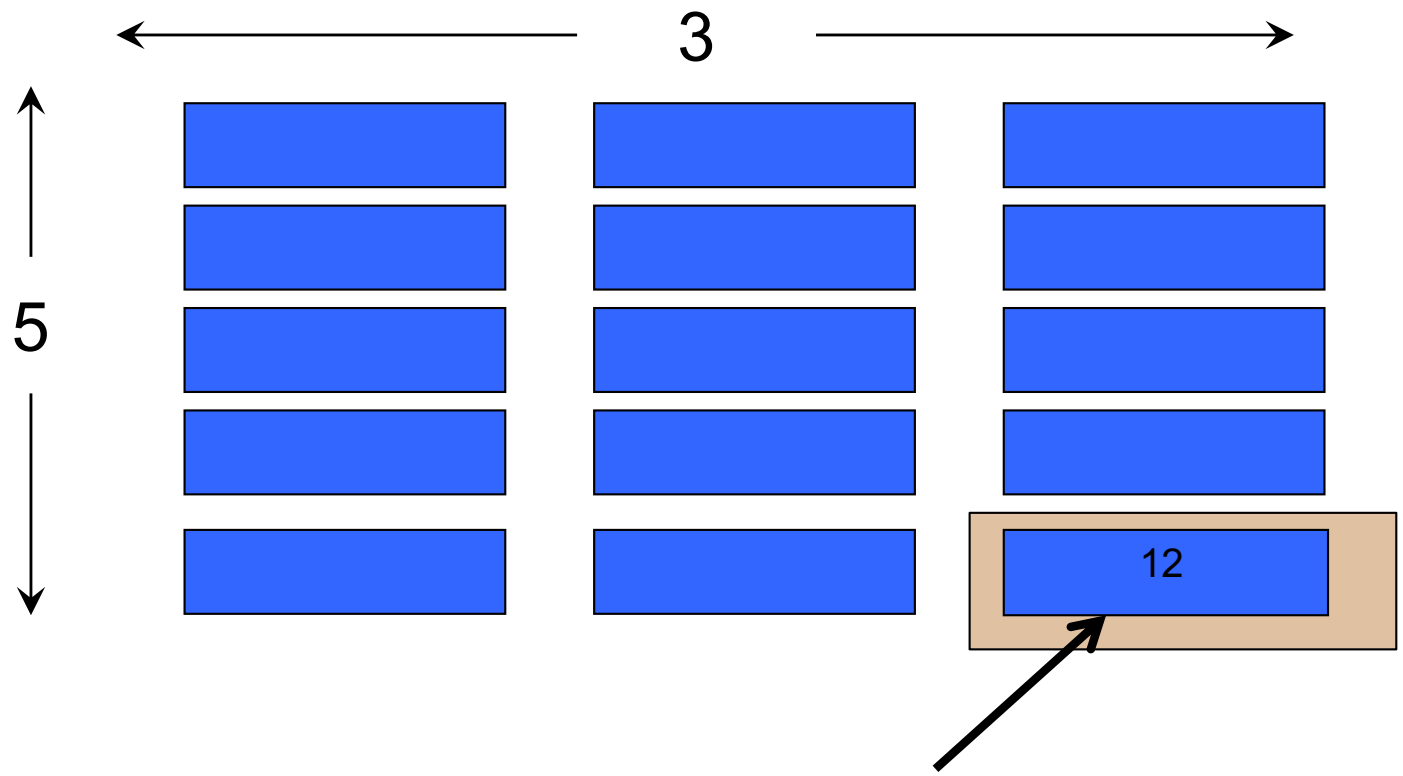
Rank = 1

Dimension = 10

HDF5 Datatype

- Describe individual data elements in an HDF5 dataset
- Wide range of datatypes supported
  - Integer
  - Float
  - Enum
  - Array
  - User-defined (e.g., 13-bit integer)
  - Variable length types (e.g., strings)
  - Compound (similar to C structs)
  - Many more …

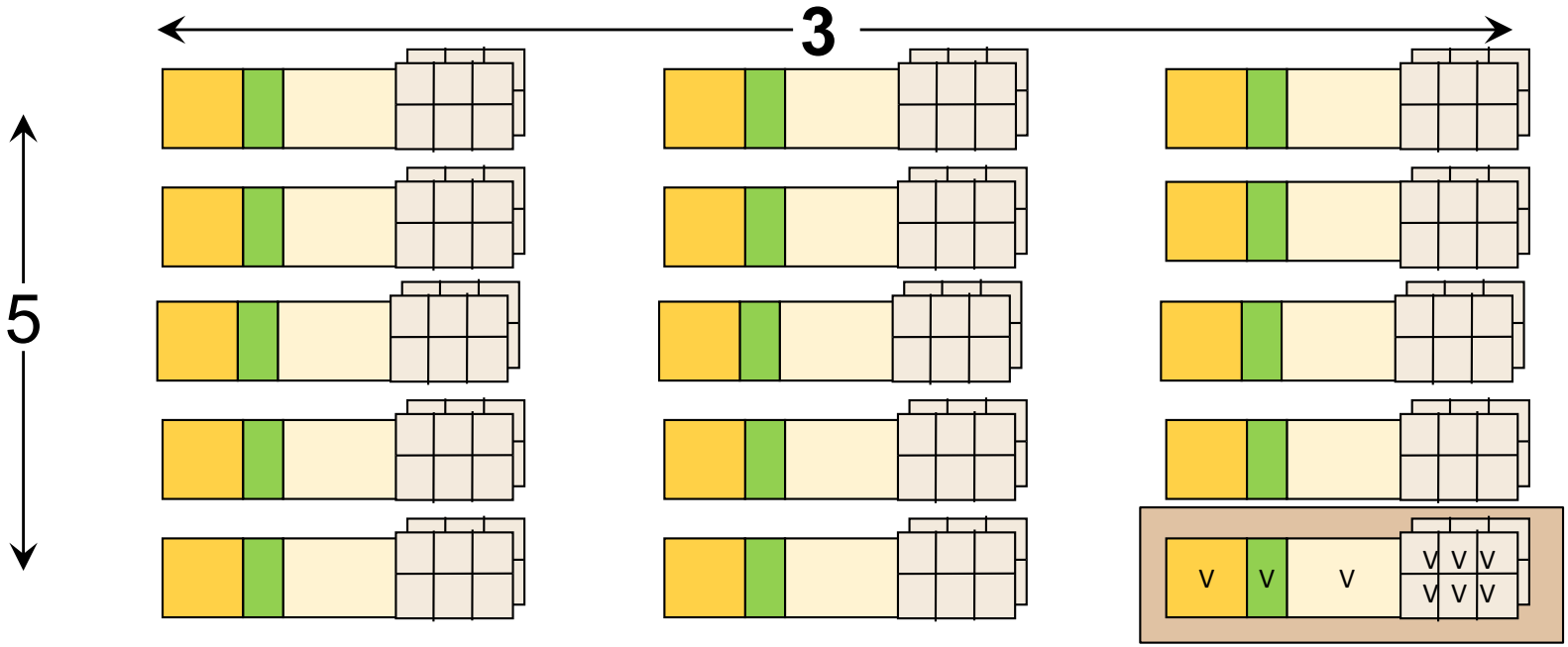
HDF5 Dataset



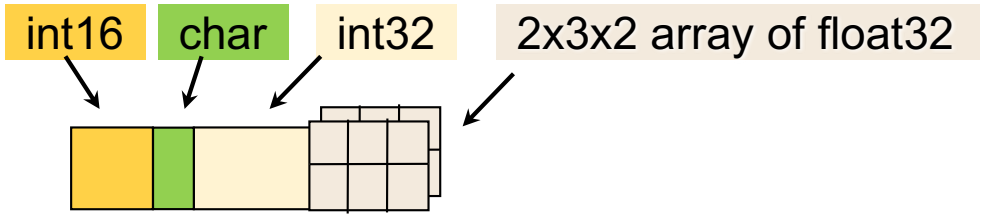
Datatype: 32-bit Integer

Dataspace: Rank = 2
Dimensions = 5 x 3

HDF5 Dataset with Compound Datatype

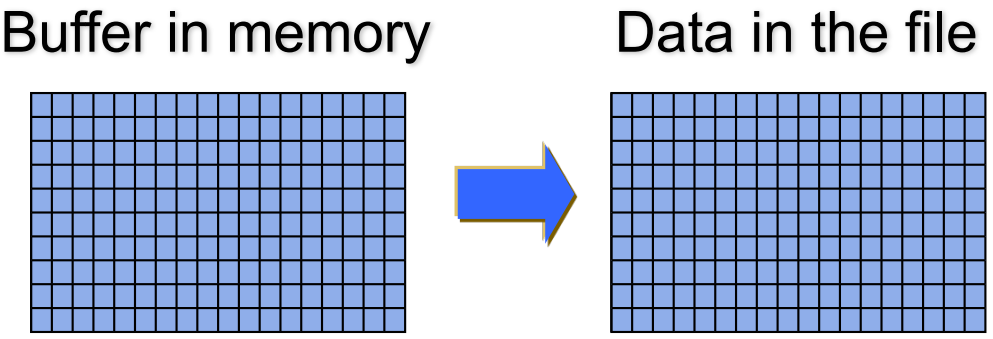


Compound Datatype:



Dataspace: Rank = 2
Dimensions = 5 x 3

How data is stored?



Contiguous (default)

Data elements stored physically adjacent to each other

Chunked

Better access time for subsets; extendible

Chunked & Compressed

Improves storage efficiency, transmission speed

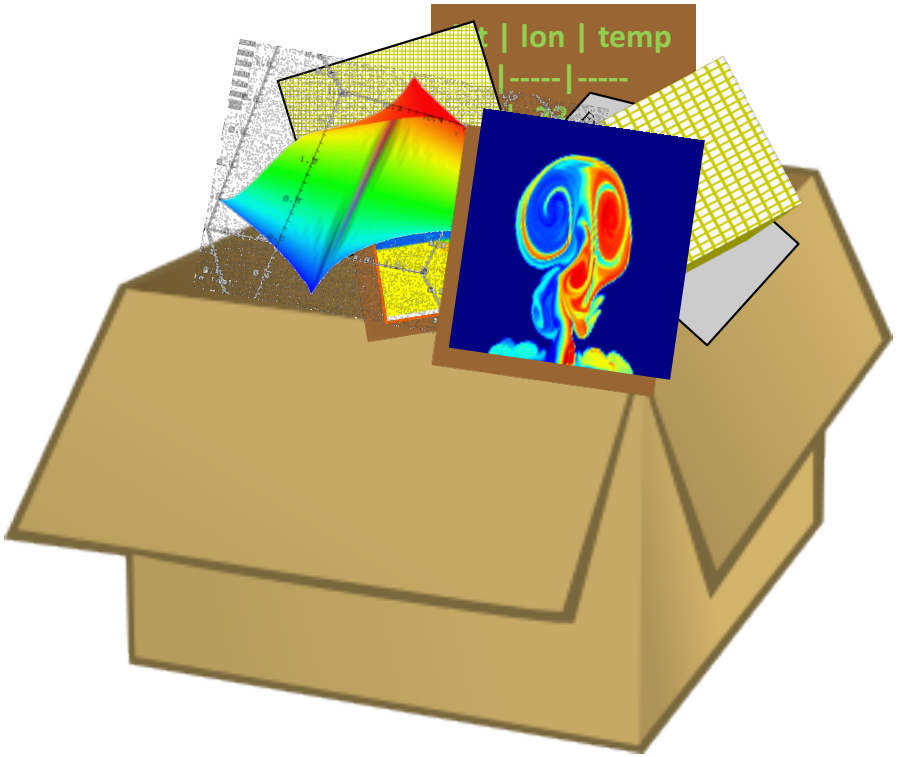
HDF5 Attributes

- Typically contain user metadata
- Have a name and a value
- Attributes "decorate" HDF5 objects
- Value is described by a datatype and a dataspace

  Analogous to a dataset, but do not support partial I/O operations; nor can they be compressed or extended

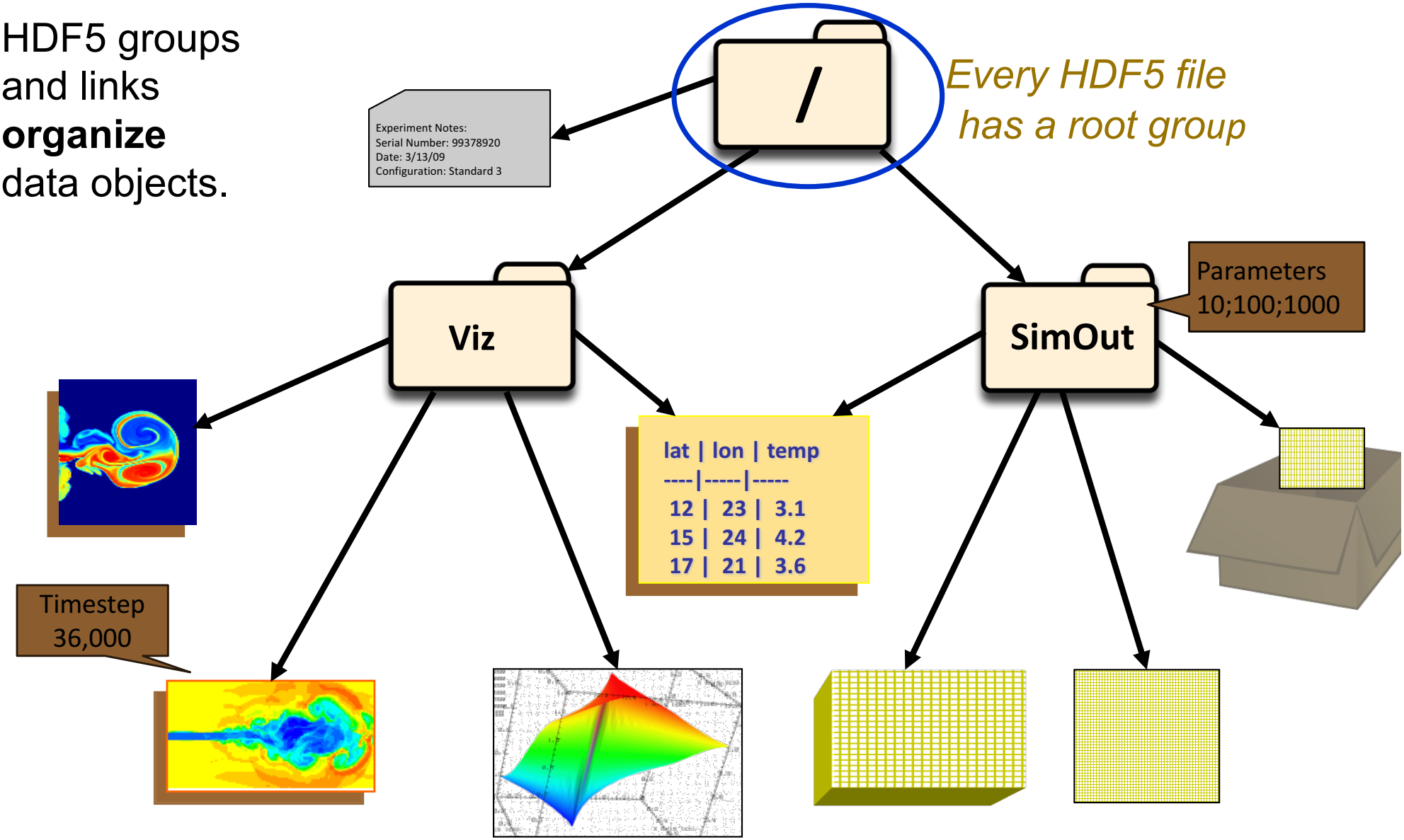
HDF5 File

An HDF5 file is a **smart container** that holds data objects.

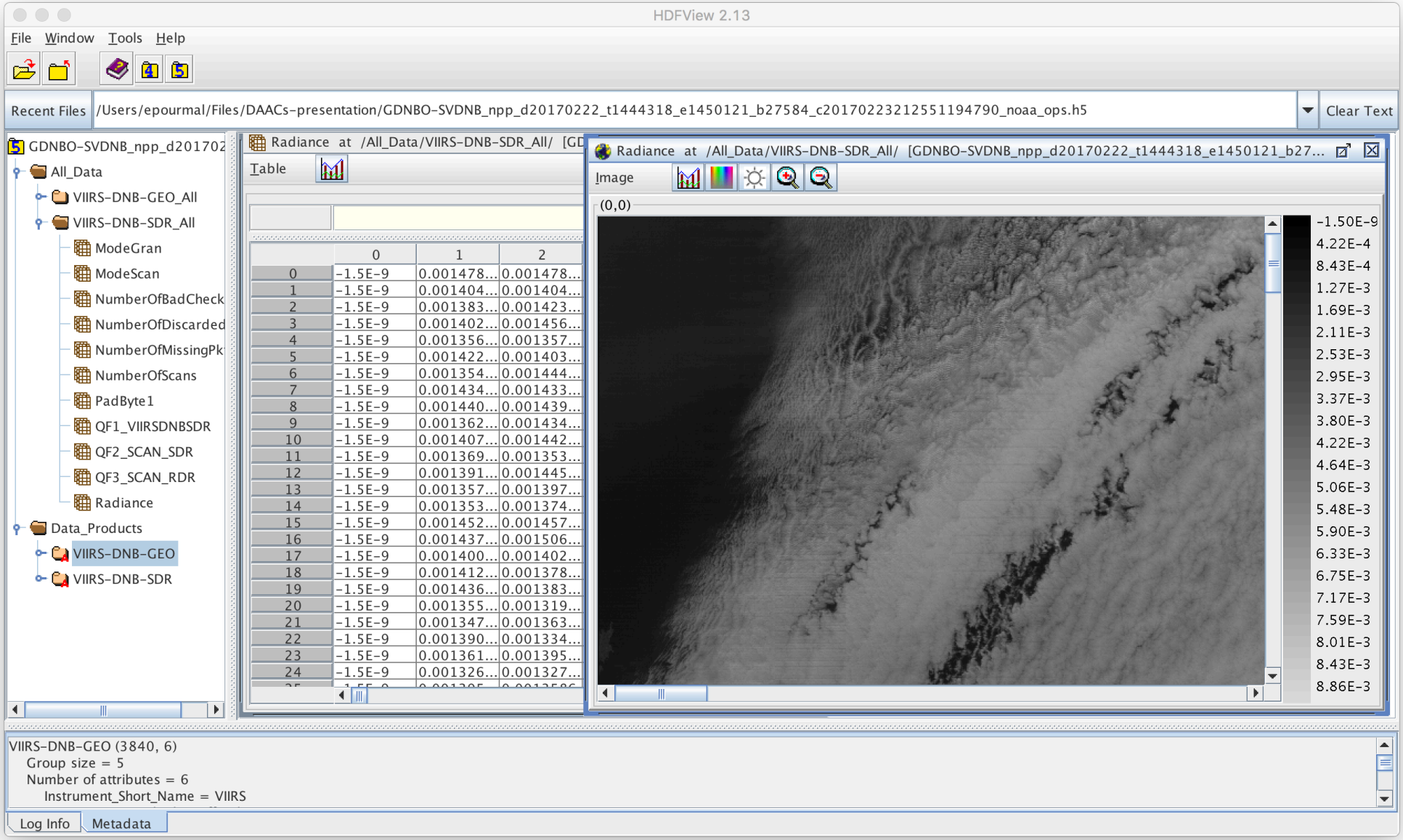


HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



HDFView: Example of HDF5 file



HDF5 SOFTWARE

HDF5 Home Page

HDF5 home page: http://hdfgroup.org/HDF5/

- Latest release: HDF5 1.8.21, HDF5 1.10.4

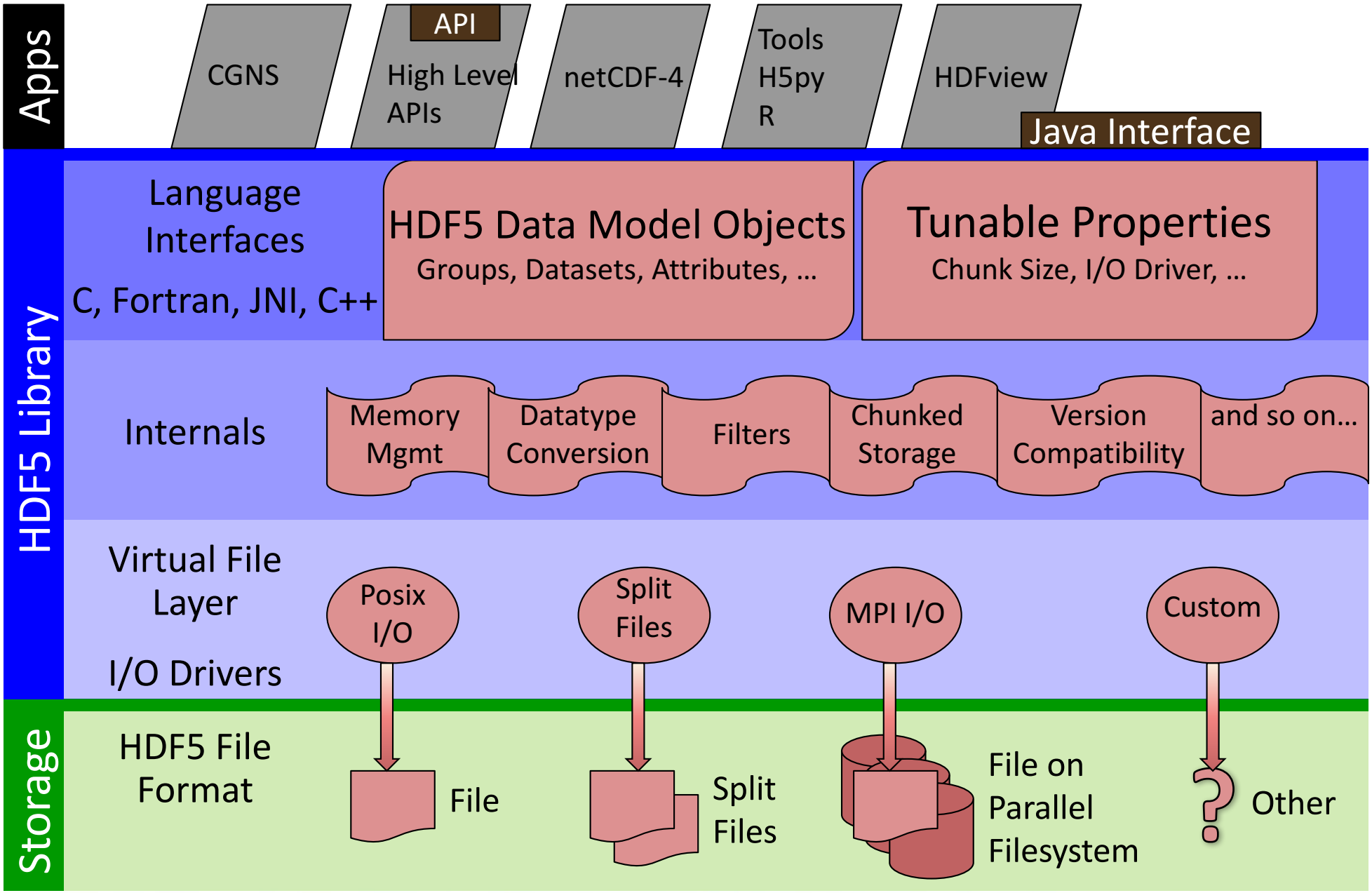
HDF5 source code:

- Written in C, and includes optional C++, Fortran, Java APIs, and High-Level APIs
- Contains command-line utilities (h5dump, h5repack, h5diff, ..) and compile scripts

HDF5 pre-built binaries:

- When possible, include C, C++, F90, and High-Level libraries. Check ./lib/libhdf5.settings file.
- Built with and require the SZIP and ZLIB external libraries

HDF5 Software Layers and Storage



Useful Tools For New Users

`h5dump`:

Tool to "dump" or display contents of HDF5 files

`h5cc, h5c++, h5fc`:

Scripts to compile applications

`HDFView`:

Java browser to view HDF5 files

https://portal.hdfgroup.org/display/support/Download+HDFView

HDF5 Examples (C, Fortran, Java, Python, Matlab)

https://portal.hdfgroup.org/display/HDF5/HDF5

HDF5 PROGRAMMING MODEL AND API

General Programming Paradigm

- Object is opened or created
- Object is accessed, possibly many times
- Object is closed

- Properties of object are optionally defined
  - ✓Creation properties (e.g., use chunking storage)
  - ✓Access properties (e.g., parallel I/O)

Order of Operations

- An order is imposed on operations by argument dependencies

  For 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.

The General HDF5 API

- C, Fortran, Java, C++, and .NET bindings
- IDL, MATLAB, Python (H5Py, PyTables)
- C routines begin with prefix H5*?*

  ? is a character corresponding to the type of object the function acts on

Example Functions:

H5D :	**D**ataset interface	*e.g.,* **H5Dread**	**h5dread_f**
H5F :	**F**ile interface	*e.g.,* **H5Fopen**	**h5dopen_f**
H5S :	data**S**pace interface	*e.g.,* **H5Sclose**	**h5sclose_f**

The HDF5 API

- For flexibility, the API is extensive
  - ✓ 500+ functions

Victorinox
Swiss Army
Cybertool 34

- This can be daunting… but there is hope
  - ✓ A few functions can do a lot
  - ✓ Start simple
  - ✓ Build up knowledge as more features are needed

Basic Functions

H5**F**create (H5**F**open)	*create (open) File*
H5**S**create_simple/H5**S**create	*create dataSpace*
H5**D**create (H5**D**open)	*create (open) Dataset*
H5**D**read, H5**D**write	*access Dataset*
H5**D**close	*close Dataset*
H5**S**close	*close dataSpace*
H5**F**close	*close File*

Other Common Functions

DataSpaces:	H5Sselect_hyperslab (Partial I/O) H5Sselect_elements (Partial I/O) H5Dget_space
Groups:	H5Gcreate, H5Gopen, H5Gclose
Attributes:	H5Acreate, H5Aopen_name, H5Aclose, H5Aread, H5Awrite
Property lists:	H5Pcreate, H5Pclose H5Pset_chunk, H5Pset_deflate

How to compile HDF5 applications

Autotools build

Wrappers contain compiler/linker flags

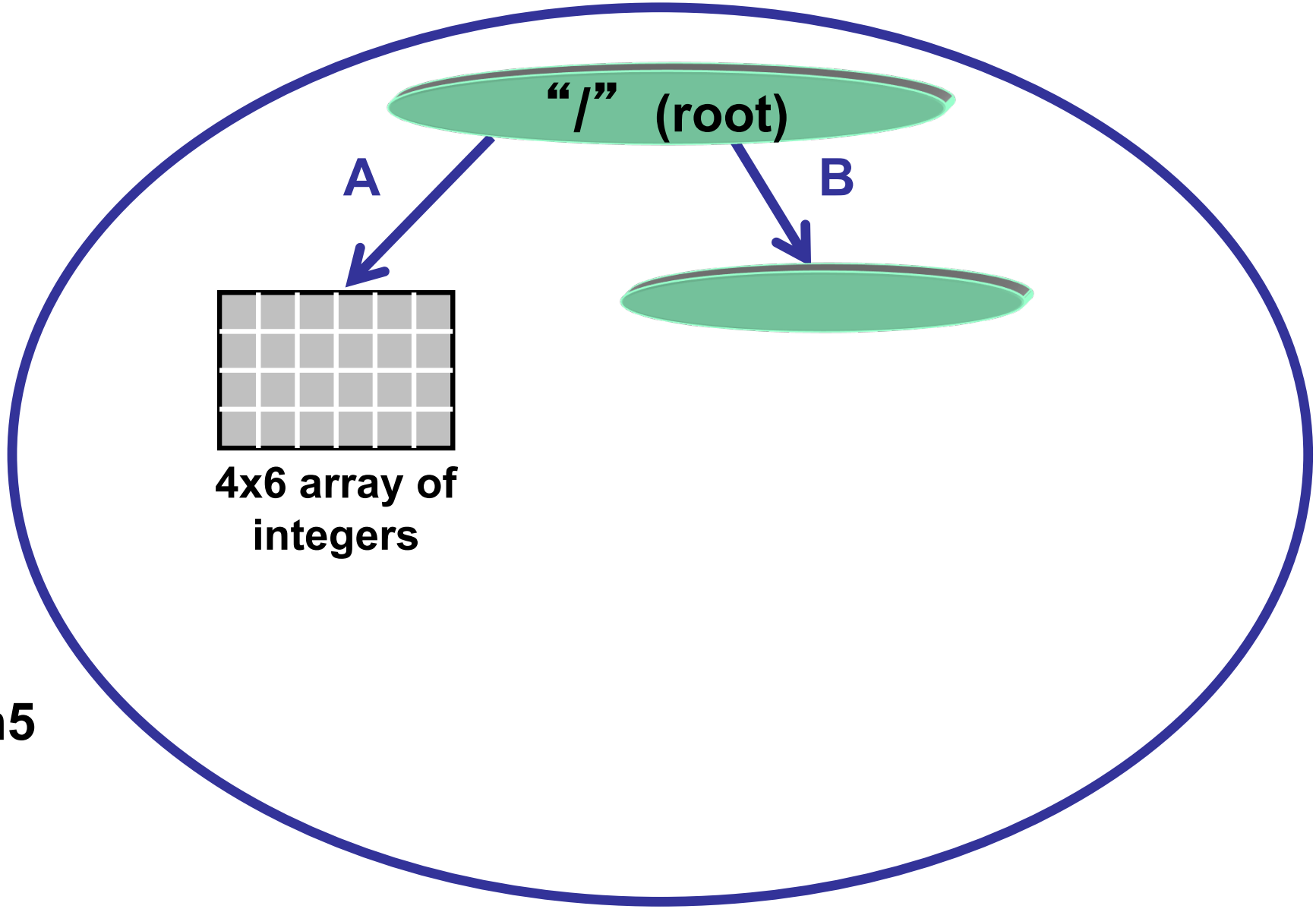
- **h5cc** – HDF5 C compiler command
- **h5fc** – HDF5 F90 compiler command
- **H5c++** – HDF5 C++ compiler command

- To compile:

```
h5cc h5prog.c
h5fc h5prog.f90
h5c++ h5prog.cpp
```

FORTRAN EXAMPLES

Content of file.h5



h5dump of file.h5

```
$ h5dump file.h5
HDF5 "file.h5" {
GROUP "/" {
   DATASET "A" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 6, 4 ) / ( 6, 4 ) }
      DATA {
      (0,0): 1, 7, 13, 19,
      (1,0): 2, 8, 14, 20,
      (2,0): 3, 9, 15, 21,
      (3,0): 4, 10, 16, 22,
      (4,0): 5, 11, 17, 23,
      (5,0): 6, 12, 18, 24
      }
   }
   GROUP "B" {
   }
}
}
```

Fortran Examples Demo 1

- Create a file `file.h5`

  `h5_crtfile.f90`
- Create a file `file.h5 and` a group `/B` in it

  `h5_crtgrp.f90`
- Open the file and create a dataset `/A`

  `h5_crtdat.f90`
- Open the file and dataset and write data to the dataset

  `h5_rdwt.f90`
- Create a table (compound dataset)

Notes on h5dump output

- Objects are list in alphabetical order
  - Creation order can be tracked (later)
- Datatype is shown as `H5T_STD_I32LE` instead of `H5T_NATIVE_INTEGER`
  - `h5dump` shows "self-described" type
- Data is shown transformed
  - Discussed in detail in the next topic "HDF5 Fortran Interface"

Thank You!
Questions?