



HDF5 Fortran Interface

The HDF Group
October 16, 2018



Goal

- Provide an overview of HDF5 Fortran library and how to use it
 - Fortran vs. C storage order
 - HDF5 Fortran APIs
 - HDF5 F2003 programming model
 - Fortran language types vs. HDF5 datatypes



Fortran vs. C Storage Order



C vs. Fortran Storage Order

h5_rdwt.f90 in Demo-1 directory writes 2-dim array (4,6)

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24



C vs. Fortran Storage Order (cont'd)

```
$ 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
        ...
      }
    }
  }
}
```

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

h5dump shows transpose data !



C vs. Fortran Storage Order (cont'd)

- HDF5 C library never touches applications' buffer
- Data will be stored as in the application buffer
 - Column order (Fortran)
 - Row order (C)
- HDF5 command-line tools are C tools and display data by row
 - When a C application reads data stored from a Fortran program, the data will appear to be transposed



C vs. Fortran Storage Order

Fastest changing dimension
↓

Fortran storage is by column

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

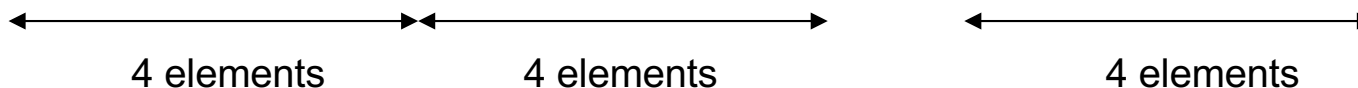
C storage is by row

1	7	13	19
2	8	14	20
3	9	15	21
4	10	16	22
5	11	17	23
6	12	18	24



Fastest changing dimension
→

1	7	13	19	3	9	..	..	23	6	12	18	24
---	---	----	----	---	---	----	----	----	---	----	----	----





Exercise 1



Example C vs. Fortran

- In EX-1 directory:
 - Run C example `h5ex_d_rdwr.c`

```
% h5cc h5ex_d_rdwr.c
```

```
% ./a.out
```
 - Copy the file `h5ex_d_rdwr.h5`

```
% cp h5ex_d_rdwr.h5 h5ex_d_rdwr.h5-C
```
 - Run Fortran example `h5ex_d_rdwr.f90`

```
% h5fc h5ex_d_rdwr.f90
```

```
% ./a.out
```
 - Use `h5dump` to compare output

```
% h5dump h5ex_d_rdwr.h5-C
```

```
% h5dump h5ex_d_rdwr.h5
```



FORTRAN APIs



Fortran Examples

- Where to find examples?

Documentation

<https://portal.hdfgroup.org/display/HDF5/Examples+by+API>

Bitbucket

<https://bitbucket.hdfgroup.org/projects/HDF5/repos/hdf5-examples/browse>

Installed code

share directory

Source code

fortran/examples directory



Documentation

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

The screenshot shows a web browser window displaying the HDF5 Support Page. The browser's address bar shows the URL <https://portal.hdfgroup.org/display/HDF5/HDF5>. The page features a dark blue header with the 'The HDF Group' logo and a search bar. Below the header is a navigation bar with links for 'Documentation', 'Knowledge', 'Help Desk', and 'Downloads'. A large blue banner with the text 'HDF5' is prominently displayed. On the left side, there is a sidebar with a tree view of the documentation structure, including links to 'Learning HDF5', 'HDF5 Examples', 'HDF5 User's Guide', 'HDF5 Application Developer's Guide', 'Design Specifications', and 'Libraries and Tools Reference'. The main content area on the right contains the heading 'Welcome to the HDF5 Support Page!' followed by 'Current Releases: HDF5-1.8.20, HDF5-1.10.3'. Below this, a paragraph describes HDF5 as a data model, library, and file format. A 'Documentation' section lists links to various guides and manuals. At the bottom right, a timestamp indicates the page was last modified on August 22, 2018.

The HDF Group

Documentation Knowledge Help Desk Downloads

HDF5

HDF5

Expand all Collapse all

- > Learning HDF5
- > HDF5 Examples
- > HDF5 User's Guide
- > HDF5 Application Developer's Guide
- > Design Specifications
- > Libraries and Tools Reference
 - > HDF5 C/Fortran Reference Manual
 - > HDF5 Fortran Library
 - HDF5 Fortran Datatypes, Flags and Utility Functions
 - HDF5 Fortran User Notes
 - HDF5 Predefined Datatypes
 - New Features F2003
 - HDF5 C++ Reference Manual
 - HDF5 Java Reference Manual
 - Java HDF5 Object Package
- > Tools

Welcome to the HDF5 Support Page!

Current Releases: [HDF5-1.8.20](#), [HDF5-1.10.3](#)

HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

Documentation

- [Learning HDF5](#)
- [HDF5 Examples](#)
- [HDF5 User's Guide](#)
- [HDF5 Application Developer's Guide](#)
- [Design Specifications](#)
- [Libraries and Tools Reference](#)

— Last Modified: August 22, 2018 | 03:11 PM

What do you think of this page?



HDF5 Fortran APIs

- Fortran APIs are organized in modules parallel to the HDF5 Interfaces.
 - **USE HDF5** (HDF5 module contains all necessary modules to run HDF5 Fortran application)
- Fortran APIs come in the form of Fortran subroutines:
 - Each Fortran subroutine name is derived from the corresponding C function name by adding "_f" to the name
 - A description of each implemented Fortran subroutine and its parameters can be found the HDF5 Reference Manual.



HDF5 Fortran APIs (cont'd)

- The number and order of the Fortran subroutine parameters may differ from the number and order of the C function parameters.
 - Two additional parameters hold the return value and an error code.
 - Subroutine parameters are listed in the following order:
 - Required input parameters
 - Output parameters, including return value and error code
 - Optional input parameters



Example

C

```
hid_t H5Dcreate(hid_t loc_id, char  
*name, hid_t type_id, hid_t space_id,  
hid_t creation_prp);
```

Fortran

```
SUBROUTINE h5dcreate_f(loc_id, name,  
type_id, space_id, dset_id, hdferr,  
creation_prp)
```

Optional parameter, may be omitted if default value is used



HDF5 F90 and F2003 APIs

- Fortran APIs were introduced in early 2000
 - Based on F90 standard
 - Support for INTEGER, REAL, CHARACTER and DOUBLE PRECISION intrinsic types only
 - No support for INTEGER(KIND=...) and REAL(KIND=...)
 - No support for derived types and enumerated types
 - No support for many HDF5 features, for example,
 - Variable-length datatypes
 - File traversal routines



HDF5 F90 and F2003 APIs (cont'd)

- HDF5 1.10.* versions have F2003 features enabled by default
- Use F2003 APIs to take advantage of HDF5 functionality and to reduce programming effort
- Fortran documentation has both F90 and F2003 signatures
 - Note: Some subroutines do not have F2003 counterparts by design



HDF5 Fortran Documentation

RFC - NewFeatures_F2003.pdf

Browse HDF5 / hdf5 - Bitb...

HDF

H5D_READ

https://portal.hdfgroup.org/display/HDF5/H5D_READ

Search

DocumentationKnowledgeHelp DeskDownloads

Contents

SummaryDescriptionExample

H5D_READ

FORTNAN

HDF5

Expand allCollapse all

> Learning HDF5

> HDF5 Examples

> HDF5 User's Guide

> HDF5 Application Developer's Guide

> Design Specifications

> Libraries and Tools Reference

> HDF5 C/Fortran Reference Manual

> Core Library

> Library

> Attributes

> Datasets

> H5D_CLOSE

> H5D_CREATE

> H5D_CREATE_ANON

> H5D_EXTEND

> H5D_FILL

> H5D_FLUSH

> H5D_GATHER

> H5D_GET_ACCESS_PLIST

> H5D_GET_CHUNK_STORAGE_SIZE

> H5D_GET_CREATE_PLIST

> H5D_GET_OFFSET

> H5D_GET_SPACE

> H5D_GET_SPACE_STATUS

> H5D_GET_STORAGE_SIZE

> H5D_GET_TYPE

> H5D_ITERATE

> H5D_OPEN

> H5D_READ

> H5D_READ_CHUNK

> H5D_REFRESH

Procedure:

H5D_READ(dataset_id, mem_type_id, mem_space_id, file_space_id, xfer_plist_id, buf)

Signature:

Fortran2003:

SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, hdferr, & mem_space_id, file_space_id, xfer_prp)
INTEGER(HID_T), INTENT(IN) :: dset_id
INTEGER(HID_T), INTENT(IN) :: mem_type_id
TYPE(C_PTR), INTENT(INOUT) :: buf
INTEGER, INTENT(OUT) :: hdferr
INTEGER(HID_T), INTENT(IN), OPTIONAL :: mem_space_id
INTEGER(HID_T), INTENT(IN), OPTIONAL :: file_space_id
INTEGER(HID_T), INTENT(IN), OPTIONAL :: xfer_prp

Fortran90:

There is no direct Fortran90 counterpart for the C function `H5Dread`. Instead, that functionality is provided by two Fortran90 subroutines:

h5dread_f	Purpose: Reads data other than variable-length data.
h5dread_vl_f	Purpose: Reads variable-length data.

SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, dims, hdferr, & mem_space_id, file_space_id, xfer_prp)
IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: dset_id ! Dataset identifier
INTEGER(HID_T), INTENT(IN) :: mem_type_id ! Memory datatype identifier
TYPE, INTENT(INOUT) :: buf ! Data buffer; may be a scalar

What do you think of this page?



HDF5 F2003 Programming Model



HDF5 Fortran Programming Model

- Each Fortran application must **call** the **h5open_f** subroutine to initialize the Fortran interface before calling any HDF5 Fortran subroutine.
- It is a good practice to **call** the **h5close_f** subroutine after all calls to the HDF5 Fortran library to close the Fortran interface.
- Fortran **indices are 1-based**



HDF5 F2003 Programming Model

- To use HDF5 F2003 interfaces follow this programming model in your application:
 - Include the `ISO_C_BINDING` module
 - Use the following declarations for variable and functions:
 - a. Use `TARGET` attribute in declarations of a variable or an array that contains data to be written or read by the HDF5 Fortran APIs
 - b. Use the `C_PTR` derived data type to declare a pointer to a variable or array in item a.
 - c. Use the `BIND(C)` attribute in derived type declaration



HDF5 F2003 Programming model (cont'd)

- To use HDF5 F2003 interfaces follow this programming model in your application:
 - d. Use the **BIND(C)** attribute in a Fortran callback declaration
 - e. **C_FUNPTR** derived type to declare a pointer to the Fortran function in item d.
- Associate a pointer with a variable or an array using **C_LOC** intrinsic datatype, and then pass it to an HDF5 Fortran call
- Associate a pointer with a callback function using the **C_FUNCTION** intrinsic data type, and then pass it to an HDF5 Fortran call.



Example: Passing a buffer

```
PROGRAM main
```

```
  USE ISO_C_BINDING
```

```
  USE HDF5
```

```
  ...
```

```
  TYPE, BIND(C) :: sensor_t
```

```
    ...
```

```
  END TYPE sensor_t
```

```
  TYPE(sensor_t), DIMENSION(1:100), TARGET :: wdata ! Write buffer
```

```
  TYPE(C_PTR) :: ptr
```

```
  ...
```

```
  ptr = C_LOC(wdata(1))
```

```
  CALL h5dwrite_f(dset, memtype, ptr, hdferr)
```

```
  ...
```

```
END PROGRAM main
```



Fortran 2003: Passing a buffer

In Demo-1 directory:

- Modify `h5_rdwt.f90` to use Fortran 2003 features to simplify `h5dwrite(read)_f` calls



Example: Passing a call back function

```
PROGRAM main
  USE ISO_C_BINDING
  USE HDF5
  ! Type iter_info and call op_func function are declared in liter_cb_mod
  USE liter_cb_mod
  ...
  TYPE(C_PTR) :: ptr
  TYPE(C_FUNPTR) :: funptr
  TYPE(iter_info), TARGET :: info
  ...
  ptr = C_LOC(info)
  funptr = C_FUNLOC(op_func)
  CALL h5literate_f(file, ..., funptr, ptr, ...)
  ...
END PROGRAM main
```



Fortran Language Types vs. HDF5 Datatypes



HDF5 Library Intrinsic Types

- See “[HDF5 Fortran Datatypes, Flags and Utility Functions](#)” document.
- For portability HDF5 C library and Fortran wrappers libraries define intrinsic types for parameters declarations; examples:

C type	Fortran type
hid_t	INTEGER(HID_T)
hsize_t	INTEGER(HSIZE_T)
size_t	INTEGER(SIZE_T)



Fortran types vs. HDF5 datatypes

- HDF5 Fortran library provides predefined HDF5 datatypes in form of handles (cmp. with hard coded integers in HDF4) that correspond to the Fortran intrinsic types

Fortran type	HDF5 Fortran type
INTEGER	H5T_NATIVE_INTEGER
REAL	H5T_NATIVE_REAL
CHARACTER	H5T_NATIVE_CHARACTER

- *There are no HDF5 predefined types for kinds of INTEGERS and REALS*
 - Use h5kind_to_type function to get HDF5 datatype



Example of using h5kind_to_type

```
INTEGER(SELECTED_INT_KIND (5)), DIMENSION(100), TARGET ::  
ivar
```

```
...
```

```
mem_type = h5kind_to_type(KIND(ivar(1)), H5_INTEGER_KIND)
```

```
ptr = C_LOC(ivar(1))
```

```
CALL h5dwrite_f (dset_t, mem_type, ptr, error)
```

- See h5ex_d_rdwr_kind_F03.f90 in EX-1 directory
- Modify examples h5_crtdat.f90 and h5_rdwt.f90 in Demo-1 to use INTEGER(KIND=8)



Fortran types vs. HDF5 datatypes

- More on working with Fortran types as we go
 - Derived types (HDF5 compound types)
 - Enum type (HDF5 enum type)
 - Strings
 - Variable-length types



Thank You!



The HDF Group



Questions/comments?