* Vgroups (V API)

# Chapter Overview

This chapter describes the vgroup data model and the Vgroup interface (also called the V interface or the V API). The first section describes the vgroup data model. The second section introduces the Vgroup interface, followed by a presentation of a programming model for vgroups. The next three sections describe the use of the Vgroup interface in accessing and creating vgroups. The final two sections cover vgroup attributes and obsolete Vgroup interface routines.

## The Vgroup Data Model

A vgroup is a structure designed to associate related data objects. The general structure of a vgroup is similar to that of the UNIX file system in that the vgroup may contain references to other vgroups or HDF data objects just as the UNIX directory may contain subdirectories or files (see Figure 5a). In previous versions of HDF, the data objects in a vgroup were limited to vdatas and vgroups. The data objects that belong to a vgroup are often referred to as the vgroup’s members.

* Similarity of the HDF Vgroup Structure and the UNIX File System

### Vgroup Names and Classes



A vgroup can have a name and/or a class associated with it. The vgroup name and class are useful in describing and classifying the data objects belonging to the vgroup.

A vgroup name is a character string and is used to semantically distinguish between vgroups in an HDF file. Multiple vgroups in a file can have the same name; however, unique names make it easier to distinguish among vgroups and are recommended.

A vgroup class is a character string and can be used to classify data objects by their intended use. For example, a vdata object named "Storm Tracking Data - 5/11/94" and another vdata object named "Storm Tracking Data - 6/23/94" can be grouped together under a vgroup named "Storm Tracking Data - 1994". If the data was collected in Anchorage, Alaska the class name might be "Anchorage Data", particularly if other vgroups contain storm track data collected in different locations.

The specific use of the vgroup name and class name is solely determined by HDF users.

### Vgroup Organization

There are many ways to organize vgroups through the use of the Vgroup interface. Vgroups may contain any number of vgroups and data objects, including data objects and vgroups that are members of other vgroups. Therefore, a data object may have more than one parent vgroup. For example, Data object A and Vgroup B, shown in Figure 5b, are members of multiple vgroups with different organizational structures.

* Sharing Data Objects among Vgroups

A vgroup can contain any combination of data objects. Figure 5c illustrates a vgroup that contains two raster images and a vdata.



* A Vgroup Containing Two 8-Bit Raster Images, or RIS8 Objects, and a Vdata

### An Example Using Vgroups



Although vgroups can contain any combination of HDF data objects, it is often useful to establish conventions on the content and structure of vgroups. This section, with the illustration in Figure 5d, describes an example of a vgroup convention that is used by scientific and graphics programmers to describe the surfaces of a mathematical or material object as well as its properties.

This vgroup consists of one list of coordinate data, one list of connectivity data, and one list of node property data. These three lists are stored in separate vdata objects within the vgroup.

Each 2-dimensional coordinate in the list of coordinate data defines the relative location of a vertex, or node. Each entry in the list of connectivity data is an ordered list of node numbers which describes a polygon. This ordered list is referred to as the connectivity list. For example, the number "2" as an item in a connectivity list would represent the second entry in the node table. Node properties are user-defined values attached to each node within the polygon and can be numbers or characters.

For example, consider a heated mesh of 400 triangles formed by connecting 1000 nodes. A vgroup describing this mesh might contain the coordinates of the vertices, the temperature value of the vertices, and a connectivity list describing the edges of the triangles.

* Vgroup Structure Describing a Heated Mesh

## The Vgroup Interface



The Vgroup interface consists of routines for creating and accessing vgroups, and getting information about vgroups and their members.

### Vgroup Interface Routines

Vgroup interface routine names are prefaced by "V" in C and by "vf" in FORTRAN-77. These routines are categorized as follows:

* Access/Create routines control access to the Vgroup interface and to individual vgroups.
* Manipulation routines modify vgroups’ characteristics, and add and delete vgroups’ members.
* Vgroup inquiry routines obtain information about vgroups. Some of these routines are useful for locating vgroups in a file.
* Member inquiry routines obtain information about members of vgroups.
* Attributes routines provide information about vgroups’ attributes.

The Vgroup interface routines are listed in Table 5A below and described in the following sections.

* Vgroup Interface Routines

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Routine Name | | Description |
| C | FORTRAN-77 |
| Access/Create | Vstart | vfstart | Initializes the Vdata and Vgroup interfaces (Section 5.4.1 on page 215) |
| Vattach | vfatch | Establishes access to a vgroup (Section 5.4.1 on page 215) |
| Vdetach | vfdtch | Terminates access to a vgroup (Section 5.4.2 on page 216) |
| Vend | vfend | Terminates access to the Vdata and Vgroup interfaces (Section 5.4.2 on page 216) |
| Manipulation | VHmakegroup | vhfmkgp | Builds a vgroup containing elements specified by their tags/refs (Section 5.5.4 on page 224) |
| Vaddtagref | vfadtr | Adds an HDF data object to a vgroup (Section 5.5.2 on page 220) |
| Vdelete | vdelete | Removes a vgroup from a file (Section 5.7.1 on page 248) |
| Vdeletetagref | vfdtr | Detaches a member from a vgroup (Section 5.7.2 on page 248) |
| Vinsert | vfinsrt | Adds a vgroup or vdata to an existing vgroup (Section 5.5.3 on page 223) |
| Vsetclass | vfscls | Assigns a class name to a vgroup (Section 5.5.1 on page 220) |
| Vsetname | vfsnam | Assigns a name to a vgroup (Section 5.5.1 on page 220) |
| Vgroup Inquiry | Vfind | vfind | Returns the reference number of a vgroup given its name (Section 5.6.1.9 on page 236) |
| Vfindclass | vfndcls | Returns the reference number of a vgroup specified by class name (Section 5.6.1.10 on page 236) |
| Vgetclass | vfgcls | Retrieves the class of a vgroup (Section 5.6.1.7 on page 235) |
| Vgetclassnamelen | [unavailable] | Retrieves the length of a vgroup’s class name (Section 5.6.1.8 on page 235) |
| Vgetid | vfgid | Returns the reference number for the next vgroup in the HDF file (Section 5.6.1.2 on page 232) |
| Vgetname | vfgnam | Retrieves the name of a vgroup (Section 5.6.1.5 on page 234) |
| Vgetnamelen | [unavailable] | Retrieves the length of a vgroup’s name (Section 5.6.1.6 on page 235) |
| Vgetversion | vfgver | Returns the vgroup version of a vgroup (Section 5.8.1 on page 249) |
| Vinquire | vfinq | Retrieves general information about a vgroup (Section 5.9.2 on page 258) |
| Vlone | vflone | Retrieves the reference numbers of vgroups that are not members of other vgroups (Section 5.6.1.1 on page 231) |
| Vntagrefs | vfntr | Returns the number of tag/reference number pairs contained in the specified vgroup (Section 5.6.2.1 on page 240) |
| VQueryref | vqref | Returns the reference number of a vgroup (Section 5.6.2.9 on page 247) |
| VQuerytag | vqtag | Returns the tag of a vgroup (Section 5.6.2.10 on page 247) |
| Member Inquiry | Vflocate | vffloc | Locates a vdata in a vgroup given a list of field names (Section 5.6.2.7 on page 246) |
| Vgetnext | vfgnxt | Returns the identifier of the next vgroup or vdata in a vgroup (Obsolete) (Section 5.9.1 on page 258) |
| Vgettagref | vfgttr | Retrieves a tag/reference number pair for a data object in the vgroup (Section 5.6.2.2 on page 240) |
| Vgettagrefs | vfgttrs | Retrieves the tag/reference number pairs of all of the data objects belonging to a vgroup (Section 5.6.2.3 on page 241) |
| Vinqtagref | vfinqtr | Determines whether a data object belongs to a vgroup (Section 5.6.2.4 on page 245) |
| Visvg | vfisvg | Determines whether a data object is a vgroup within another vgroup (Section 5.6.2.5 on page 245) |
| Visvs | vfisvs | Determines whether a data object is a vdata within a vgroup (Section 5.6.2.6 on page 246) |
| Vnrefs | vnrefs | Retrieves the number of tags of a given tag type in a vgroup (Section 5.6.2.8 on page 246) |
| Attributes | Vattrinfo | vfainfo | Retrieves information of a vgroup attribute (Section 5.8.5 on page 251) |
| Vfindattr | vffdatt | Returns the index of a vgroup attribute given the attribute name (Section 5.8.3 on page 249) |
| Vgetattr | vfgnatt/  vfgcatt | Retrieves the values of a vgroup attribute (Section 5.8.7 on page 253) |
| Vnattrs | vfnatts | Returns the total number of vgroup attributes (Section 5.8.4 on page 250) |
| Vsetattr | vfsnatt/  vfscatt | Sets the attribute of a vgroup (Section 5.8.2 on page 249) |

### Identifying Vgroups in the Vgroup Interface

The Vgroup interface identifies vgroups in several ways. In some cases, a vgroup can be accessed directly through the use of its unique reference number. In other cases, the reference number and the routine Vattach are used to obtain a vgroup identifier. The reference number of a vgroup can be obtained from the name or the class of the vgroup, or by sequentially traversing the file. The concept of reference number is discussed in Section 2.2.2.1 on page 8.

When a vgroup is attached or created, it is assigned an identifier, called vgroup id. After a vgroup has been attached or created, its identifier is used by the Vgroup interface routines in accessing the vgroup.

## Programming Model for the Vgroup Interface

The programming model for accessing vgroups is as follows:

* Open an HDF file.
* Initialize the Vgroup interface.
* Create a new vgroup or open an existing one.
* Perform the desired operations on the vgroup.
* Terminate access to the vgroup.
* Terminate access to the Vgroup interface.
* Close the file.

These steps correspond to the following sequence of function calls:

C: file\_id = Hopen(filename, file\_access\_mode, num\_dds\_block);

status = Vstart(file\_id);

vgroup\_id = Vattach(file\_id, vgroup\_ref, vg\_access\_mode);

<Optional operations>

status = Vdetach(vgroup\_id);

status = Vend(file\_id);

status = Hclose(file\_id);

FORTRAN: file\_id = hopen(filename, file\_access\_mode, num\_dds\_block)

status = vfstart(file\_id)

vgroup\_id = vfatch(file\_id, vgroup\_ref, vg\_access\_mode)

<Optional operations>

status = vfdtch(vgroup\_id)

status = vfend(file\_id)

status = hclose(file\_id)

The calling program must obtain a separate vgroup identifier for each vgroup to be accessed.

### Accessing Files and Vgroups: Vstart and Vattach

An HDF file must be opened by Hopen before it can be accessed using the Vgroup interface. Hopen is described in Chapter 2, HDF Fundamentals.

The Vgroup interface routines are used in a similar manner to the Vdata interface routines. Before performing operations on a vgroup, a calling program must call Vstart for every file to be accessed. Vstart initializes the internal vgroup structures in a file. Vstart takes one argument, the file identifier returned by Hopen, and returns either SUCCEED (or 0) or FAIL (or -1). Note that the Vstart routine is used by both the Vdata and Vgroup interfaces.

The calling program must also call one Vattach for every vgroup to be accessed. Vattach provides access to an individual vgroup for all read and write operations. Vattach takes three arguments: file\_id, vgroup\_ref, and vg\_access\_mode, and returns either a vgroup identifier or FAIL (or -1).

The argument file\_id is the file identifier returned by Hopen. The parameter vgroup\_ref is the reference number that identifies the vgroup to be accessed. Specifying vgroup\_ref with a value of -1 will create a new vgroup; specifying vgroup\_ref with a nonexistent reference number will return an error code of FAIL (or -1); and specifying vgroup\_ref with a valid reference number will initiate access to the corresponding vgroup.

When a new vgroup is created, it does not have any members. Additional operations must be performed to add other HDF data objects to the vgroup. Refer to Section 5.5 on page 218 for information.

To access an existing vdata, its reference number must be obtained. The Vgroup interface includes two routines for this purpose, Vfindand Vgetid. Vfind can be used to obtain the reference number of a vgroup when the vgroup’s name is known. Vgetid can be used to obtain the reference number by sequentially traversing the file. These routines are discussed in Section 5.6.1.9 on page 236 and Section 5.6.1.2 on page 232.

The parameter vg\_access\_mode in Vattach specifies the type of access ("r" or "w") required for operations on the selected vgroup.

Multiple attaches may be made to a vgroup, which will result in several vgroup identifiers being assigned to the same vgroup. Termination must be properly handled as described in the next section.

The parameters of Vstart and Vattach are defined in Table 5B on page 216.

### Terminating Access to Vgroups and Files: Vdetach and Vend

Successfully terminating access to a vgroup requires one Vdetach call for every Vattach call made. Similarly, successfully terminating access to the Vgroup interface requires one Vend call for every Vstart call made.

Vdetach terminates access to a vgroup by updating internal library structures and freeing all memory associated with the vgroup and allocated by Vattach. Once a vgroup is detached, its identifier is invalid and any attempts to access this vgroup identifier will result in an error condition. Vdetach takes one argument, vgroup\_id, the vgroup identifier returned by Vattach, and returns either SUCCEED (or 0) or FAIL (or -1).

Vend releases all internal data structures allocated by Vstart. Attempts to use the Vgroup interface identifier after calling Vend will produce errors. Vend takes one argument, file\_id, the file identifier returned by Hopen, and returns either SUCCEED (or 0) or FAIL (or -1). Note that the first Vend call to a file must occur after all Vdetach calls for the vgroups in the same file have been made. Note also that the Vend routine is used by both the Vdata and Vgroup interfaces.

Hclose must be used to terminate access to the HDF file and only after all proper Vend calls are made. Hclose is described in Chapter 2, HDF Fundamentals.

The parameters of Vdetach and Vend are also defined in Table 5B.

* Vstart, Vattach, Vdetach, and Vend Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vstart  [intn]  (vfstart) | file\_id | int32 | integer | File identifier |
| Vattach  [int32]  (vfatch) | file\_id | int32 | integer | File identifier |
| vgroup\_ref | int32 | integer | Reference number for an existing vgroup or -1 to create a new one |
| vg\_access\_mode | char \* | character\*(\*) | Access mode of the vgroup operation |
| Vdetach  [int32]  (vfdtch) | vgroup\_id | int32 | integer | Vgroup identifier |
| Vend  [intn]  (vfend) | file\_id | int32 | integer | File identifier |

* Creating HDF Files and Vgroups

This example illustrates the use of Hopen/hopen, Vstart/vfstart, Vattach/vfatch, Vdetach/vfdtch, Vend/vfend, and Hclose/hclose to create and to access two vgroups in an HDF file.

The program creates the HDF file, named "Two\_Vgroups.hdf", and two vgroups stored in the file. Note that, in this example, the program only create two empty vgroups.

C version

C:

#include "hdf.h"

#define FILE\_NAME "Two\_Vgroups.hdf"

main()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n; /\* returned status for functions returning an intn \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

vgroup\_ref = -1,

vgroup1\_id, vgroup2\_id, file\_id;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Create the HDF file.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_CREATE, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Create the first vgroup. Note that the vgroup reference number is set

\* to -1 for creating and the access mode is "w" for writing.

\*/

vgroup1\_id = Vattach (file\_id, vgroup\_ref, "w");

/\*

\* Create the second vgroup.

\*/

vgroup2\_id = Vattach (file\_id, vgroup\_ref, "w");

/\*

\* Any operations on the vgroups.

\*/

/\*

\* Terminate access to the first vgroup.

\*/

status\_32 = Vdetach (vgroup1\_id);

/\*

\* Terminate access to the second vgroup.

\*/

status\_32 = Vdetach (vgroup2\_id);

/\*

\* Terminate access to the V interface and close the HDF file.

\*/

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

}

FORTRAN-77 version

FORTRAN:

program create\_vgroup

implicit none

C

C Parameter declaration

C

character\*15 FILE\_NAME

C

parameter (FILE\_NAME = ’Two\_Vgroups.hdf’)

integer DFACC\_CREATE

parameter (DFACC\_CREATE = 4)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfdtch, vfend

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status

integer file\_id

integer vgroup1\_id, vgroup2\_id, vgroup\_ref

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Create the HDF file.

C

file\_id = hopen(FILE\_NAME, DFACC\_CREATE, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Create the first vgroup. Note that the vgroup reference number is set

C to -1 for creating and the access mode is ’w’ for writing.

C

vgroup\_ref = -1

vgroup1\_id = vfatch(file\_id, vgroup\_ref, ’w’)

C

C Create the second vgroup.

C

vgroup2\_id = vfatch(file\_id, vgroup\_ref, ’w’)

C

C Any operations on the vgroups.

C

C ..............................

C

C Terminate access to the first vgroup.

C

status = vfdtch(vgroup1\_id)

C

C Terminate access to the second vgroup.

C

status = vfdtch(vgroup2\_id)

C

C Terminate access to the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

## Creating and Writing to a Vgroup

There are two steps involved in the creation of a vgroup: creating the vgroup and inserting data objects into it. Any HDF data object can be inserted into a vgroup. Creation and insertion operations are usually performed at the same time, but that is not required.

HDF provides two routines that insert an HDF data object into a vgroup, Vaddtagref and Vinsert. Vaddtagref can insert any HDF data object into a vgroup, but requires that the tag and reference number of the object be available. Refer to Section 2.2.2.1 on page 8 for the description of tags and reference numbers for HDF data objects. Vinsert only inserts a vdata or a vgroup to a vgroup, but only requires the identifier of the vdata or the vgroup.

Creating a vgroup with a member involves the following steps:

* Open the HDF file.
* Initialize the Vgroup interface.
* Create the new vgroup.
* Optionally assign a vgroup name.
* Optionally assign a vgroup class.
* Insert a data object.
* Terminate access to the vgroup.
* Terminate access to the Vgroup interface.
* Close the HDF file.

These steps correspond to the following sequence of function calls:

C: file\_id = Hopen(filename, file\_access\_mode, num\_dds\_block);

status = Vstart(file\_id);

vgroup\_id = Vattach(file\_id, vgroup\_ref, vg\_access\_mode);

status = Vsetname(vgroup\_id, vgroup\_name);

status = Vsetclass(vgroup\_id, vgroup\_class);

/\* Use either Vinsert to add a vdata or a vgroup, or

Vaddtagref to add any data object \*/

num\_of\_tag\_refs = Vaddtagref(vgroup\_id, obj\_tag, obj\_ref);

OR obj\_pos = Vinsert(vgroup\_id, v\_id);

status = Vdetach(vgroup\_id);

status = Vend(file\_id);

status = Hclose(file\_id);

FORTRAN: file\_id = hopen(filename, file\_access\_mode, num\_dds\_block)

status = vfstart(file\_id)

vgroup\_id = vfatch(file\_id, vgroup\_ref, vg\_access\_mode)

status = vfsnam(vgroup\_id, vdata\_name)

status = vfscls(vgroup\_id, vdata\_class)

C Use either Vinsert to add a vdata or a vgroup, or Vaddtagref to

C add any data object

num\_of\_tag\_refs = vfadtr(vgroup\_id, obj\_tag, obj\_ref)

OR obj\_pos = vfinsrt(vgroup\_id, v\_id)

status = vfdtch(vgroup\_id)

status = vfend(file\_id)

status = hclose(file\_id)

The parameter v\_id in the calling sequence is either a vdata or vgroup identifier. The parameter vgroup\_id is the vgroup identifier returned by Vattach.

When a new vgroup is created, the value of vgroup\_ref must be set to -1 and the value of vg\_access\_mode must be "w".

### Assigning a Vgroup Name and Class: Vsetname and Vsetclass

Vsetname assigns a name to a vgroup. The parameter vgroup\_name is a character string with the name to be assigned to the vgroup. If Vsetname is not called, the vgroup name is set to a zero-length character string. A name may be assigned and reset any time after the vgroup is created.

Vsetclass assigns a class to a vgroup. The parameter vgroup\_class is a character string with the class name to be assigned to the vgroup. If Vsetclass is not called, the vgroup class is set to a zero-length string. As with the vgroup names, the class may be set and reset at any time after the vgroup is created.

Starting from release 4.2.4, the maximum length of vgroup’s name is no longer limited to VGNAMELENMAX (or 64) and release 4.2.5 for vgroup’s class name.

Vsetname and Vsetclass return either SUCCEED (or 0) or FAIL (or -1). The parameters of these routines are further described in Table 5C on page 225.

### Inserting Any HDF Data Object into a Vgroup: Vaddtagref

Vaddtagref inserts HDF data objects into the vgroup identified by vgroup\_id. HDF data objects may be added to a vgroup when the vgroup is created or at any point thereafter.

The parameters obj\_tag and obj\_ref in Vaddtagref are the tag and reference number, respectively, of the data object to be inserted into the vgroup. Note that duplicated tag and reference number pairs are allowed.

Vaddtagref returns the total number of tag and reference number pairs, i.e., the total number of data objects, in the vgroup if the operation is successful, and FAIL (or -1) otherwise. The parameters of Vaddtagref are further described in Table 5C.

Note that Vaddtagref does not verify that the tag and reference number exist.

* Adding an SDS to a New Vgroup

This example illustrates the use of Vaddtagref/vfadtr to add an HDF data object, an SDS specifically, to a vgroup.

In this example, the program first creates the HDF file "General\_Vgroups.hdf", then an SDS in the SD interface, and a vgroup in the Vgroup interface. The SDS is named "Test SD" and is a one-dimensional array of type int32 of 10 elements. The vgroup is named "SD Vgroup" and is of class "Common Vgroups". The program then adds the SDS to the vgroup using Vaddtagref/vfadtr. Notice that, when the operations are complete, the program explicitly terminates access to the SDS, the vgroup, the SD interface, and the Vgroup interface before closing the HDF file. Refer to Chapter 3, Scientific Data Sets (SD API) for the discussion of the SD routines used in this example.



C version

C:

#include "hdf.h" /\* Note: in this example, hdf.h can be omitted...\*/

#include "mfhdf.h" /\* ...since mfhdf.h already includes hdf.h \*/

#define FILE\_NAME "General\_Vgroups.hdf"

#define SDS\_NAME "Test SD"

#define VG\_NAME "SD Vgroup"

#define VG\_CLASS "Common Vgroups"

main()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n; /\* returned status for functions returning an intn \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

sd\_id, /\* SD interface identifier \*/

sds\_id, /\* data set identifier \*/

sds\_ref, /\* reference number of the data set \*/

dim\_sizes[1], /\* dimension of the data set - only one \*/

rank = 1, /\* rank of the data set array \*/

vgroup\_id, /\* vgroup identifier \*/

file\_id; /\* HDF file identifier, same for V interface \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Create the HDF file.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_CREATE, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Initialize the SD interface.

\*/

sd\_id = SDstart (FILE\_NAME, DFACC\_WRITE);

/\*

\* Set the size of the SDS’s dimension.

\*/

dim\_sizes[0] = 10;

/\*

\* Create the SDS.

\*/

sds\_id = SDcreate (sd\_id, SDS\_NAME, DFNT\_INT32, rank, dim\_sizes);

/\*

\* Create a vgroup and set its name and class.

\*/

vgroup\_id = Vattach (file\_id, -1, "w");

status\_32 = Vsetname (vgroup\_id, VG\_NAME);

status\_32 = Vsetclass (vgroup\_id, VG\_CLASS);

/\*

\* Obtain the reference number of the SDS using its identifier.

\*/

sds\_ref = SDidtoref (sds\_id);

/\*

\* Add the SDS to the vgroup. Note: the tag DFTAG\_NDG is used

\* when adding an SDS. Refer to Appendix A for the entire list of tags.

\*/

status\_32 = Vaddtagref (vgroup\_id, DFTAG\_NDG, sds\_ref);

/\*

\* Terminate access to the SDS and to the SD interface.

\*/

status\_n = SDendaccess (sds\_id);

status\_n = SDend (sd\_id);

/\*

\* Terminate access to the vgroup and to the V interface, and

\* close the HDF file.

\*/

status\_32 = Vdetach (vgroup\_id);

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

}

FORTRAN-77 version

FORTRAN:

program add\_SDS\_to\_a\_vgroup

implicit none

C

C Parameter declaration

C

character\*19 FILE\_NAME

character\*7 SDS\_NAME

character\*9 VG\_NAME

character\*13 VG\_CLASS

C

parameter (FILE\_NAME = ’General\_Vgroups.hdf’,

+ SDS\_NAME = ’Test SD’,

+ VG\_NAME = ’SD Vgroup’,

+ VG\_CLASS = ’Common Vgroups’)

integer DFACC\_CREATE, DFACC\_WRITE

parameter (DFACC\_CREATE = 4, DFACC\_WRITE = 2)

integer DFNT\_INT32

parameter (DFNT\_INT32 = 24)

integer DFTAG\_NDG

parameter (DFTAG\_NDG = 720)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfsnam, vfscls, vfadtr, vfdtch, vfend

integer sfstart, sfcreate, sfid2ref, sfendacc, sfend

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status

integer file\_id

integer vgroup\_id

integer sd\_id, sds\_id, sds\_ref

integer dim\_sizes(1), rank

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Create the HDF file.

C

file\_id = hopen(FILE\_NAME, DFACC\_CREATE, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Initialize SD interface.

C

sd\_id = sfstart(FILE\_NAME, DFACC\_WRITE)

C

C Set the rank and the size of SDS’s dimension.

C

rank = 1

dim\_sizes(1) = 10

C

C Create the SDS.

C

sds\_id = sfcreate(sd\_id, SDS\_NAME, DFNT\_INT32, rank, dim\_sizes)

C

C Create a vgroup and set its name and class.

C

vgroup\_id = vfatch(file\_id, -1 , ’w’)

status = vfsnam(vgroup\_id, VG\_NAME)

status = vfscls(vgroup\_id, VG\_CLASS)

C

C Obtain the reference number of the SDS using its identifier.

C

sds\_ref = sfid2ref(sds\_id)

C

C Add the SDS to the vgroup. Note: the tag DFTAG\_NDG is used

C when adding an SDS. Refer to HDF Reference Manual, Section III, Table 3K,

C for the entire list of tags.

C

status = vfadtr(vgroup\_id, DFTAG\_NDG, sds\_ref)

C

C Terminate access to the SDS and to the SD interface.

C

status = sfendacc(sds\_id)

status = sfend(sd\_id)

C

C Terminate access to the vgroup.

C

status = vfdtch(vgroup\_id)

C

C Terminate access to the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

### Inserting a Vdata or Vgroup Into a Vgroup: Vinsert

Vinsert is a routine designed specifically for inserting vdatas or vgroups into a parent vgroup. To use Vinsert, you must provide the identifier of the parent vgroup, vgroup\_id, as well as the identifier of the vdata or vgroup to be inserted, v\_id.

The parameter v\_id of Vinsert is either a vdata identifier or a vgroup identifier, depending on whether a vdata or vgroup is to be inserted.

Vinsert returns the index of the inserted vdata or vgroup if the operation is successful, and FAIL (or -1) otherwise. The parameters of Vinsert are further defined in Table 5C.

### Building a Vgroup with or without Elements: VHmakegroup

VHmakegroup is a high-level routine, designed to facilite the process of creating and inserting elements into a vgroup. The vgroup will have a name and/or class name if these information are provided to VHmakegroup. By using VHmakegroup, an application can by pass a number of function calls such as Vattach, Vsetname, Vsetclass, Vinsert/Vaddtagref, and Vdetach.

VHmakegroup creates a vgroup with the name specified by the parameter vgroup\_name and the class name specified by the parameter vgroup\_class in the file identified by the parameter file\_id. The routine inserts n\_objects objects into the vgroup. The tag and reference numbers of the objects to be inserted are specified in the arrays tag\_array and ref\_array*.*

Creating empty vgroups with VHmakegroup is allowed. VHmakegroup does not check if the tag/reference number pair is valid, or if the corresponding data object exists. However, all of the tag/reference number pairs must be unique.

Vstart must precede any calls to VHmakegroup.

The elements in the arrays tag\_array and ref\_arrayare the matching tag/reference number pairs of the objects to be inserted, that means tag\_array[0] and ref\_array[0] refer to one data object, and tag\_array[1] and ref\_array[1] to another, etc. If name and/or class name are not desired, the parameters vgroup\_name and/or vgroup\_class can be NULL.

The syntax of VHmakegroup is as follows:

C: vgroup\_ref = VHmakegroup(file\_id, tag\_array, ref\_array, n\_objects, vgroup\_name, vgroup\_class);

FORTRAN: vgroup\_ref = vhfmkgp(file\_id, tag\_array, ref\_array, n\_objects, vgroup\_name, vgroup\_class)

VHmakegroup returns the reference number of the newly-created vgroup if successful, FAIL (or -1) otherwise.

The parameters of VHmakegroup are further defined in Table 5F.

* Vsetname, Vsetclass, Vaddtagref, Vinsert, and VHmakegroup Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vsetname  [int32]  (vfsnam) | vgroup\_id | int32 | integer | Vgroup identifier |
| vgroup\_name | char \* | character\*(\*) | Vgroup name |
| Vsetclass  [int32]  (vfscls) | vgroup\_id | int32 | integer | Vgroup identifier |
| vgroup\_class | char \* | character\*(\*) | Vgroup class |
| Vaddtagref  [int32]  (vfadtr) | vgroup\_id | int32 | integer | Vgroup identifier |
| obj\_tag | int32 | integer | Tag of the data object to be inserted |
| obj\_ref | int32 | integer | Reference number of the data object to be inserted |
| Vinsert  [int32]  (vfinsrt) | vgroup\_id | int32 | integer | Vgroup identifier |
| v\_id | int32 | integer | Identifier of the vgroup or vdata to be inserted |
| VHmakegroup  [int32]  (vhfmkgp) | file\_id | int32 | integer | File identifier |
| tag\_array | int32 \* | integer(\*) | Array of tags |
| ref\_array | int32 \* | integer(\*) | Array of reference numbers |
| n\_objects | int32 | integer | Number of items in tag\_array or ref\_array (must be the same) |
| vgroup\_name | char \* | character\*(\*) | Name of the vgroup |
| vgroup\_class | char \* | character\*(\*) | Class of the vgroup |

* Adding Three Vdatas into a Vgroup

This example illustrates the use of Vinsert/vfinsrt to add a vdata to a vgroup. Note that Vaddtagref/vfadtrf, used in the previous example, performs the same task and only differs in the argument list.

In this example, the program creates three vdatas and a vgroup in the existing HDF file "General\_Vgroups.hdf" then adds the three vdatas to the vgroup. Notice that the vdatas and the vgroup are created in the same interface that is initialized by the call Vstart/vfstart. The first vdata is named "X,Y Coordinates" and has two order-1 fields of type float32. The second vdata is named "Temperature" and has one order-1 field of type float32. The third vdata is named "Node List" and has one order-3 field of type int16. The vgroup is named "Vertices" and is of class "Vertex Set". The program uses Vinsert/vfinsrt to add the vdatas to the vgroup using the vdata identifiers. Refer to Chapter 3, Vdatas (VS API), for the discussion of the VS routines used in this example.



C version

C:

#include "hdf.h"

#define FILE\_NAME "General\_Vgroups.hdf"

#define N\_RECORDS 30 /\* number of records in the vdatas \*/

#define ORDER 3 /\* order of field FIELD\_VD2 \*/

#define VG\_NAME "Vertices"

#define VG\_CLASS "Vertex Set"

#define VD1\_NAME "X,Y Coordinates" /\* first vdata to hold X,Y...\*/

#define VD1\_CLASS "Position" /\*...values of the vertices \*/

#define VD2\_NAME "Temperature" /\* second vdata to hold the...\*/

#define VD2\_CLASS "Property List" /\*...temperature field \*/

#define VD3\_NAME "Node List" /\* third vdata to hold...\*/

#define VD3\_CLASS "Mesh" /\*...the list of nodes \*/

#define FIELD1\_VD1 "PX" /\* first field of first vdata - X values \*/

#define FIELD2\_VD1 "PY"/\* second field of first vdata - Y values \*/

#define FIELD\_VD2 "TMP"/\* field of third vdata \*/

#define FIELD\_VD3 "PLIST"/\* field of second vdata \*/

#define FIELDNAME\_LIST "PX,PY" /\* field name list for first vdata \*/

/\* Note that the second and third vdatas can use the field names as

the field name lists unless more fields are added to a vdata.

Then a field name list is needed for that vdata \*/

main( )

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n; /\* returned status for functions returning an intn \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

file\_id, vgroup\_id,

vdata1\_id, vdata2\_id, vdata3\_id;

int32 num\_of\_records, /\* number of records actually written \*/

vd\_index; /\* position of a vdata in the vgroup \*/

int8 i, j, k = 0;

float32 pxy[N\_RECORDS][2] = /\* buffer for data of the first vdata \*/

{-1.5, 2.3, -1.5, 1.98, -2.4, .67,

-3.4, 1.46, -.65, 3.1, -.62, 1.23,

-.4, 3.8, -3.55, 2.3, -1.43, 2.44,

.23, 1.13, -1.4, 5.43, -1.4, 5.8,

-3.4, 3.85, -.55, .3, -.21, 1.22,

-1.44, 1.9, -1.4, 2.8, .94, 1.78,

-.4, 2.32, -.87, 1.99, -.54, 4.11,

-1.5, 1.35, -1.4, 2.21, -.22, 1.8,

-1.1, 4.55, -.44, .54, -1.11, 3.93,

-.76, 1.9, -2.34, 1.7, -2.2, 1.21};

float32 tmp[N\_RECORDS]; /\* buffer for data of the second vdata \*/

int16 plist[N\_RECORDS][3]; /\* buffer for data of the third vdata \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Open the HDF file for writing.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_WRITE, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Buffer the data for the second and third vdatas.

\*/

for (i = 0; i < N\_RECORDS; i++)

for (j = 0; j < ORDER; j++)

plist[i][j] = ++k;

for (i = 0; i < N\_RECORDS; i++)

tmp[i] = i \* 10.0;

/\*

\* Create the vgroup then set its name and class. Note that the vgroup’s

\* reference number is set to -1 for creating and the access mode is "w" for

\* writing.

\*/

vgroup\_id = Vattach (file\_id, -1, "w");

status\_32 = Vsetname (vgroup\_id, VG\_NAME);

status\_32 = Vsetclass (vgroup\_id, VG\_CLASS);

/\*

\* Create the first vdata then set its name and class. Note that the vdata’s

\* reference number is set to -1 for creating and the access mode is "w" for

\* writing.

\*/

vdata1\_id = VSattach (file\_id, -1, "w");

status\_32 = VSsetname (vdata1\_id, VD1\_NAME);

status\_32 = VSsetclass (vdata1\_id, VD1\_CLASS);

/\*

\* Introduce and define the fields of the first vdata.

\*/

status\_n = VSfdefine (vdata1\_id, FIELD1\_VD1, DFNT\_FLOAT32, 1);

status\_n = VSfdefine (vdata1\_id, FIELD2\_VD1, DFNT\_FLOAT32, 1);

status\_n = VSsetfields (vdata1\_id, FIELDNAME\_LIST);

/\*

\* Write the buffered data into the first vdata with full interlace mode.

\*/

num\_of\_records = VSwrite (vdata1\_id, (uint8 \*)pxy, N\_RECORDS,

FULL\_INTERLACE);

/\*

\* Insert the vdata into the vgroup using its identifier.

\*/

vd\_index = Vinsert (vgroup\_id, vdata1\_id);

/\*

\* Detach from the first vdata.

\*/

status\_32 = VSdetach (vdata1\_id);

/\*

\* Create, write, and insert the second vdata to the vgroup using

\* steps similar to those used for the first vdata.

\*/

vdata2\_id = VSattach (file\_id, -1, "w");

status\_32 = VSsetname (vdata2\_id, VD2\_NAME);

status\_32 = VSsetclass (vdata2\_id, VD2\_CLASS);

status\_n = VSfdefine (vdata2\_id, FIELD\_VD2, DFNT\_FLOAT32, 1);

status\_n = VSsetfields (vdata2\_id, FIELD\_VD2);

num\_of\_records = VSwrite (vdata2\_id, (uint8 \*)tmp, N\_RECORDS,

FULL\_INTERLACE);

vd\_index = Vinsert (vgroup\_id, vdata2\_id);

status\_32 = VSdetach (vdata2\_id);

/\*

\* Create, write, and insert the third vdata to the vgroup using

\* steps similar to those used for the first and second vdatas.

\*/

vdata3\_id = VSattach (file\_id, -1, "w");

status\_32 = VSsetname (vdata3\_id, VD3\_NAME);

status\_32 = VSsetclass (vdata3\_id, VD3\_CLASS);

status\_n = VSfdefine (vdata3\_id, FIELD\_VD3, DFNT\_INT16, 3);

status\_n = VSsetfields (vdata3\_id, FIELD\_VD3);

num\_of\_records = VSwrite (vdata3\_id, (uint8 \*)plist, N\_RECORDS,

FULL\_INTERLACE);

vd\_index = Vinsert (vgroup\_id, vdata3\_id);

status\_32 = VSdetach (vdata3\_id);

/\*

\* Terminate access to the vgroup "Vertices".

\*/

status\_32 = Vdetach (vgroup\_id);

/\*

\* Terminate access to the V interface and close the HDF file.

\*/

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

}

FORTRAN-77 version

FORTRAN:

program add\_vdatas\_to\_a\_vgroup

implicit none

C

C Parameter declaration

C

character\*19 FILE\_NAME

character\*8 VG\_NAME

character\*10 VG\_CLASS

character\*15 VD1\_NAME

character\*8 VD1\_CLASS

character\*11 VD2\_NAME

character\*13 VD2\_CLASS

character\*9 VD3\_NAME

character\*4 VD3\_CLASS

C

parameter (FILE\_NAME = ’General\_Vgroups.hdf’,

+ VG\_NAME = ’Vertices’,

+ VG\_CLASS = ’Vertex Set’)

parameter (VD1\_NAME = ’X,Y Coordinates’,

+ VD2\_NAME = ’Temperature’,

+ VD3\_NAME = ’Node List’)

parameter (VD1\_CLASS = ’Position’,

+ VD2\_CLASS = ’Property List’,

+ VD3\_CLASS = ’Mesh’)

character\*2 FIELD1\_VD1

character\*2 FIELD2\_VD1

character\*3 FIELD\_VD2

character\*4 FIELD\_VD3

character\*5 FIELDNAME\_LIST

parameter (FIELD1\_VD1 = ’PX’,

+ FIELD2\_VD1 = ’PY’,

+ FIELD\_VD2 = ’TMP’,

+ FIELD\_VD3 = ’PLIST’,

+ FIELDNAME\_LIST = ’PX,PY’)

integer N\_RECORDS

parameter (N\_RECORDS = 30)

integer DFACC\_WRITE

parameter (DFACC\_WRITE = 2)

integer DFNT\_FLOAT32, DFNT\_INT16

parameter (DFNT\_FLOAT32 = 5, DFNT\_INT16 = 22)

integer FULL\_INTERLACE

parameter (FULL\_INTERLACE = 0)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfsnam, vfscls, vfinsrt, vfdtch, vfend

integer vsfatch, vsfsnam, vsfscls, vsffdef, vsfsfld,

+ vsfwrt, vsfwrtc, vsfdtch

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status

integer file\_id

integer vgroup\_id

integer vdata1\_id, vdata2\_id, vdata3\_id, vd\_index

integer num\_of\_records

integer i, j, k

real pxy(2,N\_RECORDS), tmp(N\_RECORDS)

integer plist(3,N\_RECORDS)

data pxy /-1.5, 2.3, -1.5, 1.98, -2.4, .67,

+ -3.4, 1.46, -.65, 3.1, -.62, 1.23,

+ -.4, 3.8, -3.55, 2.3, -1.43, 2.44,

+ .23, 1.13, -1.4, 5.43, -1.4, 5.8,

+ -3.4, 3.85, -.55, .3, -.21, 1.22,

+ -1.44, 1.9, -1.4, 2.8, .94, 1.78,

+ -.4, 2.32, -.87, 1.99, -.54, 4.11,

+ -1.5, 1.35, -1.4, 2.21, -.22, 1.8,

+ -1.1, 4.55, -.44, .54, -1.11, 3.93,

+ -.76, 1.9, -2.34, 1.7, -2.2, 1.21/

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Open the HDF file for writing.

C

file\_id = hopen(FILE\_NAME, DFACC\_WRITE, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Buffer the data for the third and second vdatas.

C

do 20 i = 1, N\_RECORDS

do 10 j = 1, 3

plist(j,i) = k

k = k+1

10 continue

20 continue

do 30 i = 1, N\_RECORDS

tmp(i) = (i-1) \* 10.0

30 continue

C

C Create a vgroup and set its name and class.

C Note that the vgroup’s reference number is set to -1 for creating

C and the access mode is ’w’ for writing.

C

vgroup\_id = vfatch(file\_id, -1 , ’w’)

status = vfsnam(vgroup\_id, VG\_NAME)

status = vfscls(vgroup\_id, VG\_CLASS)

C

C Create the first vdata then set its name and class. Note that the vdata’s

C reference number is set to -1 for creating and the access mode is ’w’ for

C writing.

C

vdata1\_id = vsfatch(file\_id, -1, ’w’)

status = vsfsnam(vdata1\_id, VD1\_NAME)

status = vsfscls(vdata1\_id, VD1\_CLASS)

C

C Introduce and define the fields of the first vdata.

C

status = vsffdef(vdata1\_id, FIELD1\_VD1, DFNT\_FLOAT32, 1)

status = vsffdef(vdata1\_id, FIELD2\_VD1, DFNT\_FLOAT32, 1)

status = vsfsfld(vdata1\_id, FIELDNAME\_LIST)

C

C Write the buffered data into the first vdata.

C

num\_of\_records = vsfwrt(vdata1\_id, pxy, N\_RECORDS,

+ FULL\_INTERLACE)

C

C Insert the vdata into the vgroup using its identifier.

C

vd\_index = vfinsrt(vgroup\_id, vdata1\_id)

C

C Detach from the first vdata.

C

status = vsfdtch(vdata1\_id)

C

C Create, write, and insert the second vdata to the vgroup using

C steps similar to those used for the first vdata.

C

vdata2\_id = vsfatch(file\_id, -1, ’w’)

status = vsfsnam(vdata2\_id, VD2\_NAME)

status = vsfscls(vdata2\_id, VD2\_CLASS)

status = vsffdef(vdata2\_id, FIELD\_VD2, DFNT\_FLOAT32, 1)

status = vsfsfld(vdata2\_id, FIELD\_VD2)

num\_of\_records = vsfwrt(vdata2\_id, tmp, N\_RECORDS,

+ FULL\_INTERLACE)

vd\_index = vfinsrt(vgroup\_id, vdata2\_id)

status = vsfdtch(vdata2\_id)

C

C Create, write, and insert the third vdata to the vgroup using

C steps similar to those used for the first and second vdatas.

C

vdata3\_id = vsfatch(file\_id, -1, ’w’)

status = vsfsnam(vdata3\_id, VD3\_NAME)

status = vsfscls(vdata3\_id, VD3\_CLASS)

status = vsffdef(vdata3\_id, FIELD\_VD3, DFNT\_INT16, 3)

status = vsfsfld(vdata3\_id, FIELD\_VD3)

num\_of\_records = vsfwrtc(vdata3\_id, plist, N\_RECORDS,

+ FULL\_INTERLACE)

vd\_index = vfinsrt(vgroup\_id, vdata3\_id)

status = vsfdtch(vdata3\_id)

C

C Terminate access to the vgroup ’Vertices’.

C

status = vfdtch(vgroup\_id)

C

C Terminate access to the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

## Reading from Vgroups

Reading from vgroups is more complicated than writing to vgroups. The process of reading from vgroups involves two steps: locating the appropriate vgroup and obtaining information about the member or members of a vgroup. This section describes routines that provide these functionalities.

### Locating Vgroups and Obtaining Vgroup Information

There are several routines provided for the purpose of locating a particular vgroup, each corresponding to an identifying aspect of a vgroup. These aspects include whether the vgroup has vgroups included in it, the identification of the vgroup in the file based on its reference number, and the name and class name of the vgroup. The routines are described in the following subsections.

#### Locating Lone Vgroups: Vlone

A lone vgroup is one that is not a member of any other vgroups, i.e., not linked with any other vgroups. Vlone searches the file specified by the parameter file\_id and retrieves the reference numbers of lone vgroups in the file. This routine is useful for locating unattached vgroups in a file or the vgroups at the top of a grouping hierarchy. The syntax of Vlone is as follows:

C: num\_of\_lones = Vlone(file\_id, ref\_array, maxsize);

FORTRAN: num\_of\_lones = vflone(file\_id, ref\_array, maxsize)

The parameter ref\_array is an array allocated to hold the reference numbers of the found vgroups. The argument maxsize specifies the maximum size of ref\_array. At most maxsize reference numbers will be retrieved in ref\_array. The value of max\_size, the space allocated for ref\_array, depends on how many lone vgroups are expected to be found.

To use dynamic memory instead of allocating an unnecessarily large array (i.e., one that will hold the maximum possible number of reference numbers), call Vlone twice. In the first call to Vlone, set maxsize to a small value, for example, 0 or 1, then use the returned value (the total number of lone vgroups in the file) to allocate memory for ref\_array. This array is then passed into the second call to Vlone.

Vlone returns the total number of lone vgroups or FAIL (or -1). The parameters of this routine are further defined in (See Table 5D on page 234).

#### Sequentially Searching for a Vgroup: Vgetid

Vgetid sequentially searches through an HDF file to obtain the reference number of the vgroup immediately following the vgroup specified by the reference number, vgroup\_ref. The syntax of Vgetid is as follows:

C: ref\_num = Vgetid(file\_id, vgroup\_ref);

FORTRAN: ref\_num = vfgid(file\_id, vgroup\_ref)

To initiate a search, Vgetid may be called with vgroup\_ref set to -1. Doing so returns the reference number of the first vgroup in the file. Any attempt to search past the last vgroup in a file will cause Vgetid to return a value of FAIL (or -1).

Vgetid returns a vgroup reference number or FAIL (or -1). The parameters of Vgetid are further defined in (See Table 5D on page 234).

#### Retrieving vgroups in a file or in a vgroup: Vgetvgroups

Vgetvgroups retrieves a list containing reference numbers of vgroups in a file or in a vgroup, which is identified by the parameter id. The syntax of Vgetvgroups is as follows:

C: status = Vgetvgroups(id, start\_vgroup, vgroup\_count, refarray);

FORTRAN: status = vfgvgroups(id, start\_vg, vg\_count, refarray)

The library commonly use vgroups or vdatas to store HDF objects. For example, a vgroup is used to represent an SDS and a vdata for an attribute. Vgetvgroups retrieves only the vgroups that were previously created by user applications, not those that were created by the library internally. They are referred to as user-created vgroups, for brevity.

When id is a vgroup identifier, only the immediate sub-vgroups will be retrieved; that is, the sub-vgroups will not be traversed.

The parameter vgroup\_count specifies the number of values that the refarray list can hold and can be any positive number smaller than MAX\_REF (65535). If vgroup\_count is larger than the actual number of user-created vgroups, then only the actual number of user-created vgroups will be retrieved.

The retrieval starts at the vgroup number start\_vgroup going forward in the order which the vgroups were created. For example, if there are 100 vgroups that can be retrieved, specifying start\_vgroup as 90 and vgroup\_count as 10 will retrieve the last ten vgroups. The value for start\_vgroup must be non-negative and smaller than or equal to the number of user-created vgroups, which can be obtained by invoking Vgetvgroups passing in NULL for the array refarray. This number of user-created vgroups will also allow applications to sufficiently allocate space for refarray.

* When start\_vgroup is 0, the retrieval will start at the beginning of the file or the first sub-vgroup of the specified vgroup.
* When start\_vgroup is smaller than the number of user-created vgroups in the file or the specified vgroup, Vgetvgroups will start retrieving vgroups from the vgroup number start\_vgroup.
* When start\_vgroup is greater than the number of user-created vgroups in the file or the vgroup, Vgetvgroups will return FAIL (or -1).

Following are some examples of using Vgetvgroups to get the reference numbers of vgroups in a file, assuming that the file has been opened for reading successfully:

C: /\* Call Vgetvgroups the first time to get the number of vgroups in

the file to allocate refarray \*/

n\_vgs = Vgetvgroups(file\_id, 0, 0, NULL);

/\* Allocate space to retrieve reference numbers of n\_vgs vgroups \*/

refarray = (uint16 \*)HDmalloc(sizeof(uint16)\*n\_vgs);

/\* To get all the vgroups in the file: \*/

n\_vgs = Vgetvgroups(file\_id, 0, n\_vgs, refarray);

/\* Assuming n\_vgs=100, to get the first 10 vgroups in the file: \*/

n\_vgs = Vgetvgroups(file\_id, 0, 10, refarray);

/\* Assuming n\_vgs=100, to get the last 10 vgroups in the file: \*/

n\_vgs = Vgetvgroups(file\_id, 90, 10, refarray);

Following are some examples of using Vgetvgroups to get the reference numbers of vgroups in a parent vgroup:

C: vgroup\_id = Vattach(file\_id, vgroup\_ref, "r");

/\* Call Vgetvgroups the first time to get the number of vgroups in

the parent vgroup to allocate refarray \*/

n\_vgs = Vgetvgroups(vgroup\_id, 0, 0, NULL);

/\* Allocate space to retrieve reference numbers of n\_vgs vgroups \*/

refarray = (uint16 \*)HDmalloc(sizeof(uint16)\*n\_vgs);

/\* Get all the vgroups in the parent vgroup \*/

n\_vgs = Vgetvgroups(vgroup\_id, 0, n\_vgs, refarray);

/\* Close the vgroup \*/

status = Vdetach(vgroup\_id);

Note that, in the FORTRAN-77 version, if vg\_count is -1 then the function will return the number of user-created vgroups and disregard refarray; equivalent to passing NULL for refarray in the C version.

Vgetvgroups returns the number of user-created vgroups retrieved, if successful, or FAIL (or -1), otherwise. The parameters of this routine are further defined in (See Table 5D on page 234).

#### Determining Internal Vgroup: Vgisinternal

The HDF library commonly uses vgroups and vdatas to store metadata or data for the library's own use. For examples, vgroups are used to represent SDS or raster images, and vdatas are used to store attributes or dimensions. Typically, a user is only interested in vgroups/vdatas that were created by user applications, not by the library internally. Vgisinternal allows an application to find out if a vgroup is internally created.

The syntax of Vgisinternal is as follows:

C: is\_internal = Vgisinternal(vgroup\_id);

FORTRAN: Currently unavailable

Vgisinternal checks the class name of the given vgroup against the list HDF\_INTERNAL\_VGS to determine whether the vgroup was previously created by the library instead of by a user application. The names in HDF\_INTERNAL\_VGS are included:

\_HDF\_VARIABLE ("Var0.0")

\_HDF\_DIMENSION ("Dim0.0")

\_HDF\_UDIMENSION ("UDim0.0")

\_HDF\_CDF ("CDF0.0")

GR\_NAME ("RIG0.0")

RI\_NAME ("RI0.0")

There is one special case where an internal vgroup having a null class name and a name as GR\_NAME. This should be extremely rare, yet it is a possibility.

Vgisinternal returns TRUE (1) if the inquired vgroup is one that was internally created by the library, FALSE (0) otherwise, and FAIL (-1) if failure occurs. The parameters of this routine are further defined in (See Table 5D on page 234).

* Vlone, Vgetid, Vgetvgroups, and Vgisinternal Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vlone  [int32]  (vflone) | file\_id | int32 | integer | File identifier |
| ref\_array | int32 \* | integer (\*) | Buffer for the reference numbers of lone vgroups |
| maxsize | int32 | integer | Maximum number of vgroups to store in ref\_array |
| Vgetid  [int32]  (vfgid) | file\_id | int32 | integer | File identifier |
| vgroup\_ref | int32 | integer | Reference number of the current vgroup |
| Vgetvgroups  [intn]  (vfgvgroups) | id | int32 | integer | File or vgroup identifer |
| start\_vgroup | uintn | integer | Vgroup index to start retrieving at |
| vgroup\_count | uintn | integer | Number of vgroups to be retrieved |
| refarray | int32 \* | integer (\*) | Array to hold reference numbers of retrieved vgroups |
| Vgisinternal  [intn]  (unavailable) | vgroup\_id | int32 | N/A | Vgroup identifier |
|

#### Obtaining the Name of a Vgroup: Vgetname

Vgetname retrieves the name of the vgroup identified by the parameter vgroup\_id into the parameter vgroup\_name. The syntax of Vgetname is as follows:

C: status = Vgetname(vgroup\_id, vgroup\_name);

FORTRAN: status = vfgnam(vgroup\_id, vgroup\_name)

Starting from release 4.2.4, the maximum length of vgroup’s name is no longer limited to VGNAMELENMAX (or 64). When an application attempts to read a vgroup’s name that is longer than 64 characters with an insufficient buffer, the result will be unpredictable. Applications can use Vgetnamelen to get the length of the vgroup’s name prior to calling Vgetname.

Vgetname returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5E on page 236.

#### Obtaining the Length of a Vgroup’s Name: Vgetnamelen

Vgetnamelen retrieves the length of a vgroup’s name and stores it in the parameter name\_len. The vgroup is identified by the parameter vgroup\_id . The syntax of Vgetnamelen is as follows:

C: status = Vgetnamelen(vgroup\_id, name\_len);

FORTRAN: Currently unavailable

Vgetnamelen returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5E on page 236.

#### Obtaining the Class Name of a Vgroup: Vgetclass

Vgetclass retrieves the class name of the vgroup specified by the parameter vgroup\_id into the parameter vgroup\_class. The syntax of Vgetclass is as follows:

C: status = Vgetclass(vgroup\_id, vgroup\_class);

FORTRAN: status = vfgcls(vgroup\_id, vgroup\_class)

Starting from release 4.2.5, the maximum length of vgroup’s class name is no longer limited to VGNAMELENMAX (or 64). When an application attempts to read a vgroup’s name that is longer than 64 characters with an insufficient buffer, the result will be unpredictable. Applications can use Vgetclassnamelen to get the length of the vgroup’s class name prior to calling Vgetclass.

Vgetclass returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5E.

#### Obtaining the Length of a Vgroup’s Class Name: Vgetclassnamelen

Vgetclassnamelen retrieves the length of a vgroup’s class name and stores it in the parameter classname\_len. The vgroup is identified by the parameter vgroup\_id . The syntax of Vgetclassnamelen is as follows:

C: status = Vgetclassnamelen(vgroup\_id, classname\_len);

FORTRAN: Currently unavailable

Vgetclassnamelen returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5E on page 236.

* Vgetname, Vgetnamelen, Vgetclass, and Vgetclassnamelen Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vgetname  [int32]  (vfgnam) | vgroup\_id | int32 | integer | Vgroup identifier |
| vgroup\_name | char \* | character\*(\*) | Buffer for the name of the vgroup |
| Vgetnamelen  [int32]  (unavailable) | vgroup\_id | int32 | N/A | Vgroup identifier |
| name\_len | uint16\* | N/A | Buffer for the length of the vgroup’s name |
| Vgetclass  [int32]  (vfgcls) | vgroup\_id | int32 | integer | Vgroup identifier |
| vgroup\_class | char \* | character\*(\*) | Buffer for the vgroup class |
| Vgetclassnamelen  [int32]  (unavailable) | vgroup\_id | int32 | N/A | Vgroup identifier |
| classname\_len | uint16\* | N/A | Buffer for the length of the vgroup’s class name |

#### Locating a Vgroup Given Its Name: Vfind

Vfind searches the file identified by file\_id for a vgroup with the name specified by the parameter vgroup\_name. The syntax for Vfind is as follows:

C: vgroup\_ref = Vfind(file\_id, vgroup\_name);

FORTRAN: vgroup\_ref = vfind(file\_id, vgroup\_name)

Vfind returns the reference number of the vgroup if one is found, or 0 otherwise. If more than one vgroup has the same name, Vfind will return the reference number of the first one.

The parameters of Vfind are further defined in Table 5F.

#### Locating a Vgroup Given Its Class Name: Vfindclass

Vfindclass searches the file identified by file\_id for a vgroup with the class name specified by the parameter vgroup\_class. The syntax of Vfindclass is as follows:

C: vgroup\_ref = Vfindclass(file\_id, vgroup\_class);

FORTRAN: vgroup\_ref = vfndcls(file\_id, vgroup\_class)

Vfindclass returns the reference number of the vgroup if one is found, or 0 otherwise. If more than one vgroup has the same class name, Vfindclass will return the reference number of the first one.

The parameters of Vfindclass are further defined in Table 5F.

* Vfind and Vfindclass Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vfind  [int32]  (vfind) | file\_id | int32 | integer | File identifier |
| vgroup\_name | char \* | character\*(\*) | Buffer for the name of the vgroup |
| Vfindclass  [int32]  (vfndcls) | file\_id | int32 | integer | File identifier |
| vgroup\_class | char \* | character\*(\*) | Buffer for the vgroup class |

* Obtaining Information about Lone Vgroups

This example illustrates the use of Vlone/vflone to obtain the list of reference numbers of all lone vgroups in the file and the use of Vgetname/vfgnam and Vgetclass/vfgcls to obtain the name and the class of a vgroup.

In this example, the program calls Vlone/vflone twice. The first call is to obtain the number of lone vgroups in the file so that sufficient space can be allocated; the later call is to obtain the actual reference numbers of the lone vgroups. The program then goes through the list of lone vgroup reference numbers to get and display the name and class of each lone vgroup. The file used in this example is "General\_Vgroups.hdf".

C version

C:

#include "hdf.h"

#define FILE\_NAME "General\_Vgroups.hdf"

main( )

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n; /\* returned status for functions returning an intn \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

file\_id, vgroup\_id;

int32 lone\_vg\_number, /\* current lone vgroup number \*/

num\_of\_lones = 0; /\* number of lone vgroups \*/

int32 \*ref\_array; /\* buffer to hold the ref numbers of lone vgroups \*/

char vgroup\_name[VGNAMELENMAX], vgroup\_class[VGNAMELENMAX];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Open the HDF file for reading.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_READ, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Get and print the names and class names of all the lone vgroups.

\* First, call Vlone with num\_of\_lones set to 0 to get the number of

\* lone vgroups in the file, but not to get their reference numbers.

\*/

num\_of\_lones = Vlone (file\_id, NULL, num\_of\_lones );

/\*

\* Then, if there are any lone vgroups,

\*/

if (num\_of\_lones > 0)

{

/\*

\* use the num\_of\_lones returned to allocate sufficient space for the

\* buffer ref\_array to hold the reference numbers of all lone vgroups,

\*/

ref\_array = (int32 \*) malloc(sizeof(int32) \* num\_of\_lones);

/\*

\* and call Vlone again to retrieve the reference numbers into

\* the buffer ref\_array.

\*/

num\_of\_lones = Vlone (file\_id, ref\_array, num\_of\_lones);

/\*

\* Display the name and class of each lone vgroup.

\*/

printf ("Lone vgroups in this file are:\n");

for (lone\_vg\_number = 0; lone\_vg\_number < num\_of\_lones;

lone\_vg\_number++)

{

/\*

\* Attach to the current vgroup then get and display its

\* name and class. Note: the current vgroup must be detached before

\* moving to the next.

\*/

vgroup\_id = Vattach (file\_id, ref\_array[lone\_vg\_number], "r");

status\_32 = Vgetname (vgroup\_id, vgroup\_name);

status\_32 = Vgetclass (vgroup\_id, vgroup\_class);

printf (" Vgroup name %s and class %s\n", vgroup\_name,

vgroup\_class);

status\_32 = Vdetach (vgroup\_id);

} /\* for \*/

} /\* if \*/

/\*

\* Terminate access to the V interface and close the file.

\*/

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

/\*

\* Free the space allocated by this program.

\*/

free (ref\_array);

}

FORTRAN-77 version

FORTRAN:

program getinfo\_about\_vgroup

implicit none

C

C Parameter declaration

C

character\*19 FILE\_NAME

C

parameter (FILE\_NAME = ’General\_Vgroups.hdf’)

integer DFACC\_READ

parameter (DFACC\_READ = 1)

integer SIZE

parameter(SIZE = 10)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfgnam, vfgcls, vflone, vfdtch, vfend

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status

integer file\_id

integer vgroup\_id

integer lone\_vg\_number, num\_of\_lones

character\*64 vgroup\_name, vgroup\_class

integer ref\_array(SIZE)

integer i

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Initialize ref\_array.

C

do 10 i = 1, SIZE

ref\_array(i) = 0

10 continue

C

C Open the HDF file for reading.

C

file\_id = hopen(FILE\_NAME, DFACC\_READ, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Get and print the name and class name of all lone vgroups.

C First, call vflone with num\_of\_lones set to 0 to get the number of

C lone vgroups in the file and check whether size of ref\_array is

C big enough to hold reference numbers of ALL lone groups.

C If ref\_array is not big enough, exit the program after displaying an

C informative message.

C

num\_of\_lones = 0

num\_of\_lones = vflone(file\_id, ref\_array, num\_of\_lones)

if (num\_of\_lones .gt. SIZE) then

write(\*,\*) num\_of\_lones, ’lone vgroups is found’

write(\*,\*) ’increase the size of ref\_array to hold reference ’

write(\*,\*) ’numbers of all lone vgroups in the file’

stop

endif

C

C If there are any lone groups in the file,

C

if (num\_of\_lones .gt. 0) then

C

C call vflone again to retrieve the reference numbers into ref\_array.

C

num\_of\_lones = vflone(file\_id, ref\_array, num\_of\_lones)

C

C Display the name and class of each vgroup.

C

write(\*,\*) ’Lone vgroups in the file are:’

do 20 lone\_vg\_number = 1, num\_of\_lones

C

C Attach to the current vgroup, then get and display its name and class.

C Note: the current vgroup must be detached before moving to the next.

C

vgroup\_name = ’ ’

vgroup\_class = ’ ’

vgroup\_id = vfatch(file\_id, ref\_array(lone\_vg\_number), ’r’)

status = vfgnam(vgroup\_id, vgroup\_name)

status = vfgcls(vgroup\_id, vgroup\_class)

write(\*,\*) ’Vgroup name ’ , vgroup\_name

write(\*,\*) ’Vgroup class ’ , vgroup\_class

write(\*,\*)

status = vfdtch(vgroup\_id)

20 continue

endif

C

C Terminate access to the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

### Obtaining Information about the Contents of a Vgroup

This section describes the Vgroup interface routines that allow the user to obtain various information about the contents of vgroups.

#### Obtaining the Number of Objects in a Vgroup: Vntagrefs

Vntagrefs returns the number of tag/reference number pairs (i.e., the number of vgroup members) stored in the specified vgroup. The syntax of Vntagrefs is as follows:

C: num\_of\_tagrefs = Vntagrefs(vgroup\_id);

FORTRAN: num\_of\_tagrefs = vfntr(vgroup\_id)

Vntagrefs can be used together with Vgettagrefs or Vgettagref to identify the data objects linked to a given vgroup.

Vntagrefs returns 0 or a positive number representing the number of HDF data objects linked to the vgroup if successful, or FAIL (or -1) otherwise. The parameter of Vntagrefs is further defined in Table 5G on page 241.

#### Obtaining the Tag/Reference Number Pair of a Data Object within a Vgroup : Vgettagref

Vgettagref retrieves the tag/reference number pair of a specified data object stored within the vgroup identified by the parameter vgroup\_id. The syntax of Vgettagref is as follows:

C: status = Vgettagref(vgroup\_id, index, &obj\_tag, &obj\_ref);

FORTRAN: status = vfgttr(vgroup\_id, index, obj\_tag, obj\_ref)

Vgettagref stores the tag and reference number in the parameters obj\_tag and obj\_ref, respectively. The parameter index specifies the location of the data object within the vgroup and is zero-based.

Often, this routine is called in a loop to identify the tag/reference number pair of each data object belong to a vgroup. In this case, Vntagrefs is used to obtain the loop boundary.

Vgettagref returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5G on page 241.

#### Obtaining the Tag/Reference Number Pairs of Data Objects in a Vgroup: Vgettagrefs

Vgettagrefs retrieves the tag/reference number pairs of the members of a vgroup and returns the number of pairs retrieved. The syntax of Vgettagrefs is as follows:

C: num\_of\_pairs = Vgettagrefs(vgroup\_id, tag\_array, ref\_array, maxsize);

FORTRAN: num\_of\_pairs = vfgttrs(vgroup\_id, tag\_array, ref\_array, maxsize)

Vgettagrefs stores the tags into the array tag\_array and the reference numbers into the array ref\_array. The parameter maxsize specifies the maximum number of tag/reference number pairs to return, therefore each array must be at least maxsize in size.

Vgettagrefs can be used to obtain the value of maxsize if the tag/reference number pairs for all members of the vgroup are desired. To do this, set maxsize to 1 in the first call to Vgettagrefs.

Vgettagrefs returns the number of tag/reference number pairs or FAIL (or -1). The parameters of this routine are further defined in Table 5G.

* Vntagrefs, Vgettagref, and Vgettagrefs Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vntagrefs  [int32]  (vfntr) | vgroup\_id | int32 | integer | Vgroup identifier |
| Vgettagref  [intn]  (vfgttr) | vgroup\_id | int32 | integer | Vgroup identifier |
| index | int32 | integer | Index of the tag/reference number pair to be retrieved |
| obj\_tag | int32 \* | integer | Tag of the data object |
| obj\_ref | int32 \* | integer | Reference number of the data object |
| Vgettagrefs  [int32]  (vfgttrs) | vgroup\_id | int32 | integer | Vgroup identifier |
| tag\_array | int32 [] | integer (\*) | Buffer for the returned tags |
| ref\_array | int32 [] | integer (\*) | Buffer for the returned reference numbers |
| maxsize | int32 | integer | Maximum number of tag/reference number pairs to be returned |

* Obtaining Information about the Contents of a Vgroup

This example illustrates the use of Vgetid/vfgid to get the reference number of a vgroup, Vntagrefs/vfntr to get the number of HDF data objects in the vgroup, Vgettagref/vfgttr to get the tag/reference number pair of a data object within the vgroup, and Visvg/vfisvg and Visvs/vfisvs to determine whether a data object is a vgroup and a vdata, respectively.

In the example, the program traverses the HDF file "General\_Vgroups.hdf" from the beginning and obtains the reference number of each vgroup so it can be attached. Once a vgroup is attached, the program gets the total number of tag/reference number pairs in the vgroup and displays some information about the vgroup. The information displayed includes the position of the vgroup in the file, the tag/reference number pair of each of its data objects, and the message stating whether the object is a vdata, vgroup, or neither.

C version

C:

#include "hdf.h"

#define FILE\_NAME "General\_Vgroups.hdf"

main( )

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n; /\* returned status for functions returning an intn \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

file\_id, vgroup\_id, vgroup\_ref,

obj\_index, /\* index of an object within a vgroup \*/

num\_of\_pairs, /\* number of tag/ref number pairs, i.e., objects \*/

obj\_tag, obj\_ref, /\* tag/ref number of an HDF object \*/

vgroup\_pos = 0; /\* position of a vgroup in the file \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Open the HDF file for reading.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_READ, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Obtain each vgroup in the file by its reference number, get the

\* number of objects in the vgroup, and display the information about

\* that vgroup.

\*/

vgroup\_ref = -1; /\* set to -1 to search from the beginning of file \*/

while (TRUE)

{

/\*

\* Get the reference number of the next vgroup in the file.

\*/

vgroup\_ref = Vgetid (file\_id, vgroup\_ref);

/\*

\* Attach to the vgroup for reading or exit the loop if no more vgroups

\* are found.

\*/

if (vgroup\_ref == -1) break;

vgroup\_id = Vattach (file\_id, vgroup\_ref, "r");

/\*

\* Get the total number of objects in the vgroup.

\*/

num\_of\_pairs = Vntagrefs (vgroup\_id);

/\*

\* If the vgroup contains any object, print the tag/ref number

\* pair of each object in the vgroup, in the order they appear in the

\* file, and indicate whether the object is a vdata, vgroup, or neither.

\*/

if (num\_of\_pairs > 0)

{

printf ("\nVgroup #%d contains:\n", vgroup\_pos);

for (obj\_index = 0; obj\_index < num\_of\_pairs; obj\_index++)

{

/\*

\* Get the tag/ref number pair of the object specified

\* by its index, obj\_index, and display them.

\*/

status\_n = Vgettagref (vgroup\_id, obj\_index, &obj\_tag, &obj\_ref);

printf ("tag = %d, ref = %d", obj\_tag, obj\_ref);

/\*

\* State whether the HDF object referred to by obj\_ref is a vdata,

\* a vgroup, or neither.

\*/

if (Visvg (vgroup\_id, obj\_ref))

printf (" <-- is a vgroup\n");

else if (Visvs (vgroup\_id, obj\_ref))

printf (" <-- is a vdata\n");

else

printf (" <-- neither vdata nor vgroup\n");

} /\* for \*/

} /\* if \*/

else

printf ("Vgroup #%d contains no HDF objects\n", vgroup\_pos);

/\*

\* Terminate access to the current vgroup.

\*/

status\_32 = Vdetach (vgroup\_id);

/\*

\* Move to the next vgroup position.

\*/

vgroup\_pos++;

} /\* while \*/

/\*

\* Terminate access to the V interface and close the file.

\*/

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

}

FORTRAN-77 version

FORTRAN:

program vgroup\_contents

implicit none

C

C Parameter declaration

C

character\*19 FILE\_NAME

C

parameter (FILE\_NAME = ’General\_Vgroups.hdf’)

integer DFACC\_ READ

parameter (DFACC\_READ = 1)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfgid, vntrc, vfgttr, vfisvg,

+ vfisvs, vfdtch, vfend

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status

integer file\_id

integer vgroup\_id, vgroup\_ref, vgroup\_pos

integer obj\_index, num\_of\_pairs

integer obj\_tag, obj\_ref

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Open the HDF file for reading.

C

file\_id = hopen(FILE\_NAME, DFACC\_READ, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Obtain each vgroup in the file by its reference number, get the

C number of objects in the vgroup, and display the information

C about that vgroup.

C

vgroup\_ref = -1

vgroup\_pos = 0

10 continue

C

C Get the reference number of the next vgroup in the file.

C

vgroup\_ref = vfgid(file\_id, vgroup\_ref)

C

C Attach to the vgroup or go to the end if no additional vgroup is found.

C

if(vgroup\_ref. eq. -1) goto 100

vgroup\_id = vfatch(file\_id, vgroup\_ref , ’r’)

C

C Get the total number of objects in the vgroup.

C

num\_of\_pairs = vntrc(vgroup\_id)

C

C If the vgroup contains any object, print the tag/ref number

C pair of each object in vgroup, in the order they appear in the

C file, and indicate whether the object is a vdata, vgroup, or neither.

C

if (num\_of\_pairs .gt. 0) then

write(\*,\*) ’Vgroup # ’, vgroup\_pos, ’ contains:’

do 20 obj\_index = 1, num\_of\_pairs

C

C Get the tag/ref number pair of the object specified by its index

C and display them.

C

status = vfgttr(vgroup\_id, obj\_index-1, obj\_tag, obj\_ref)

C

C State whether the HDF object referred to by obj\_ref is a vdata,

C a vgroup, or neither.

C

if( vfisvg(vgroup\_id, obj\_ref) .eq. 1) then

write(\*,\*) ’tag = ’, obj\_tag, ’ ref = ’, obj\_ref,

+ ’ <--- is a vgroup ’

else if ( vfisvs(vgroup\_id, obj\_ref) .eq. 1) then

write(\*,\*) ’tag = ’, obj\_tag, ’ ref = ’, obj\_ref,

+ ’ <--- is a vdata ’

else

write(\*,\*) ’tag = ’, obj\_tag, ’ ref = ’, obj\_ref,

+ ’ <--- neither vdata nor vgroup ’

endif

20 continue

else

write (\*,\*) ’Vgroup #’, vgroup\_pos, ’ contains no HDF objects’

endif

write(\*,\*)

vgroup\_pos = vgroup\_pos + 1

goto 10

100 continue

C

C Terminate access to the vgroup.

C

status = vfdtch(vgroup\_id)

C

C Terminate access to the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

#### Testing Whether a Data Object Belongs to a Vgroup: Vinqtagref

Vinqtagref determines whether a data object is a member of the vgroup specified by the parameter vgroup\_id. The syntax of Vinqtagref is as follows:

C: true\_false = Vinqtagref(vgroup\_id, obj\_tag, obj\_ref);

FORTRAN: true\_false = vfinqtr(vgroup\_id, obj\_tag, obj\_ref)

The data object is specified by its tag/reference number pair in the parameters obj\_tag and obj\_ref. Vinqtagref returns TRUE (or 1) if the object belongs to the vgroup, and FALSE (or 0) otherwise. The parameters of this routine are further defined in Table 5H on page 246.

#### Testing Whether a Data Object within a Vgroup is a Vgroup: Visvg

Visvg determines whether the data object specified by its reference number, obj\_ref, is a vgroup and is a member of the vgroup identified by the parameter vgroup\_id. The syntax of Visvg is as follows:

C: true\_false = Visvg(vgroup\_id, obj\_ref);

FORTRAN: true\_false = vfisvg(vgroup\_id, obj\_ref)

Visvg returns either TRUE (or 1) or FALSE (or 0). The parameters of this routine are further defined in Table 5H on page 246.

#### Testing Whether an HDF Object within a Vgroup is a Vdata: Visvs

Visvs determines whether the data object specified by its reference number, obj\_ref, is a vdata and is a member of the vgroup identified by the parameter vgroup\_id. The syntax of Visvs is as follows:

C: true\_false = Visvs(vgroup\_id, obj\_ref);

FORTRAN: true\_false = vfisvs(vgroup\_id, obj\_ref)

Visvs returns either TRUE (or 1) or FALSE (or 0). The parameters of this routine are further defined in Table 5H.

* Vinqtagref, Visvg, and Visvs Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vinqtagref  [intn]  (vfinqtr) | vgroup\_id | int32 | integer | Vgroup identifier |
| obj\_tag | int32 | integer | Tag of the data object to be queried |
| obj\_ref | int32 | integer | Reference number of the data object to be queried |
| Visvg  [intn]  (vfisvg) | vgroup\_id | int32 | integer | Vgroup identifier |
| obj\_ref | int32 | integer | Data object reference number to be queried |
| Visvs  [intn]  (vfisvs) | vgroup\_id | int32 | integer | Vgroup identifier |
| obj\_ref | int32 | integer | Data object reference number to be queried |

#### Locating a Vdata in a Vgroup Given Vdata Fields: Vflocate

Vflocate locates a vdata that belongs to the vgroup identified by the parameter vgroup\_id and contains the fields specified in the parameter fieldname\_list. The syntax of Vflocate is as follows:

C: vdata\_ref = Vflocate(vgroup\_id, fieldname\_list);

FORTRAN: vdata\_ref = vffloc(vgroup\_id, fieldname\_list)

The parameter fieldname\_list is a string of comma-separated field names containing no white space, for example, “PX,PY,PZ”. Note that a vdata must contain all of the fields specified in fieldname\_list to be qualified.

Vflocate returns the reference number of the vdata, if one is found, and FAIL (or -1) otherwise. The parameters of this routine are further defined in Table 5I.

#### Retrieving the Number of Tags of a Given Type in a Vgroup: Vnrefs

Vnrefs returns the number of tags of the type specified by the parameter tag\_type in the vgroup identified by the parameter vgroup\_id. The syntax of Vnrefs is as follows:

C: num\_of\_tags = Vnrefs(vgroup\_id, tag\_type);

FORTRAN: num\_of\_tags = vnrefs(vgroup\_id, tag\_type)

Possible values of the parameter tag\_type are defined in Appendix A of this manual. Vnrefs returns 0 or the number of tags if successful, and FAIL (or -1) otherwise. The parameters of this routine are further defined in Table 5I.

* Vflocate and Vnrefs Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vflocate  [int32]  (vffloc) | vgroup\_id | int32 | integer | Vgroup identifier |
| fieldname\_list | char \* | character\*(\*) | Buffer containing the names of the fields |
| Vnrefs  [int32]  (vnrefs) | vgroup\_id | int32 | integer | Vgroup identifier |
| tag\_type | int32 | integer | Tag type |

#### Retrieving the Reference Number of a Vgroup: VQueryref

VQueryref returns the reference number of the vgroup identified by the parameter vgroup\_id, or FAIL (or -1) if unsuccessful. The syntax of VQueryref is as follows:

C: vgroup\_ref = VQueryref(vgroup\_id);

FORTRAN: vgroup\_ref = vqref(vgroup\_id)

VQueryref is further defined in Table 5J.

#### Retrieving the Tag of a Vgroup: VQuerytag

VQuerytag returns DFTAG\_VG (or 1965), which would be the tag of the vgroup identified by the parameter vgroup\_id, or FAIL (or -1) if unsuccessful. The syntax of VQuerytag is as follows:

C: vgroup\_tag = VQuerytag(vgroup\_id);

FORTRAN: vgroup\_tag = vqtag(vgroup\_id)

VQuerytag is further defined in Table 5J.

* VQueryref and VQuerytag Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| VQueryref  [int32]  (vqref) | vgroup\_id | int32 | integer | Vgroup identifier |
| VQuerytag  [int32]  (vqtag) | vgroup\_id | int32 | integer | Vgroup identifier |

## Deleting Vgroups and Data Objects within a Vgroup

The Vgroup interface includes two routines for deletion: one deletes a vgroup from a file and the other deletes a data object from a vgroup. These routines are discussed in the following subsections.

### Deleting a Vgroup from a File: Vdelete

Vdelete removes the vgroup identified by the parameter vgroup\_id from the file identified by the parameter file\_id. The syntax of Vdelete is as follows:

C: status = Vdelete(file\_id, vgroup\_id);

FORTRAN: status = vdelete(file\_id, vgroup\_id)

This routine will remove the vgroup from the internal data structures and from the file.

Vdelete returns either SUCCEED (or 0) or FAIL (or -1). The parameters of Vdelete are further described in Table 5K on page 248.

### Deleting a Data Object from a Vgroup: Vdeletetagref

Vdeletetagref deletes the data object, specified by the parameters obj\_tag and obj\_ref, from the vgroup, identified by the parameter vgroup\_id. The syntax of Vdeletetagref is as follows:

C: status = Vdeletetagref(vgroup\_id, obj\_tag, obj\_ref);

FORTRAN: status = vfdtr(vgroup\_id, obj\_tag, obj\_ref)

Vinqtagref should be used to determine whether the tag/reference number pair exists before calling Vdeletetagref. If duplicate tag/reference number pairs are found in the vgroup, Vdeletetagref deletes the first occurrence. Vinqtagref should also be used to determine whether duplicate tag/reference number pairs exist in the vgroup.

Vdeletetagref returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further described in Table 5K.

* Vdelete and Vdeletetagref Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vdelete  [int32]  (vdelete) | file\_id | int32 | integer | File identifier |
| vgroup\_id | int32 | integer | Vgroup identifier |
| Vdeletetagref  [int32]  (vfdtr) | vgroup\_id | int32 | integer | Vgroup identifier |
| obj\_tag | int32 | integer | Tag of the data object to be deleted |
| obj\_ref | int32 | integer | Reference number of the data object to be deleted |

## Vgroup Attributes

HDF version 4.1r1 and later include the ability to assign attributes to a vgroup. The concept of attributes is fully explained in Chapter 3, Scientific Data Sets (SD API). To review briefly, an attribute has a name, a data type, a number of attribute values, and the attribute values themselves. All attribute values must be of the same data type. For example, an attribute value cannot consist of ten characters and one integer, or a character value cannot be included in an attribute value consisting of two 32-bit integers.

Any number of attributes can be assigned to a vgroup, however, each attribute name must be unique among all attributes in the vgroup.

### Obtaining the Vgroup Version Number of a Given Vgroup: Vgetversion

The structure of the vgroup has gone through several changes since HDF was first written. Determining the version of any particular vgroup is necessary as some of the older versions of vgroups do not support some of the newer features, such as attributes. Vgetversion returns the version number of the vgroup identified by the parameter vgroup\_id. The syntax of Vgetversion is as follows:

C: version\_num = Vgetversion(vgroup\_id);

FORTRAN: version\_num = vfgver(vgroup\_id)

There are three valid version numbers: VSET\_OLD\_VERSION (or 2), VSET\_VERSION (or 3), and VSET\_NEW\_VERSION (or 4).

VSET\_OLD\_VERSION is returned when the vgroup is of a version that corresponds to an HDF library version before version 3.2.

VSET\_VERSION is returned when the vgroup is of a version that corresponds to an HDF library version between versions 3.2 and 4.0 release 2.

VSET\_NEW\_VERSION is returned when the vgroup is of a version that corresponds to an HDF library version of version 4.1 release 1 or higher.

Vgetversion returns the vgroup version number if successful, and FAIL (or -1) otherwise. This routine is further defined in Table 5L.

### Setting the Attribute of a Vgroup: Vsetattr

Vsetattr attaches an attribute to the vgroup specified by the parameter vgroup\_id. The syntax of Vsetattr is as follows:

C: status = Vsetattr(vgroup\_id, attr\_name, data\_type, n\_values, attr\_values);

FORTRAN: status = vfsnatt(vgroup\_id, attr\_name, data\_type, n\_values, attr\_values)

OR status = vfscatt(vgroup\_id, attr\_name, data\_type, n\_values, attr\_values)

If the attribute with the name specified in the parameter attr\_name already exists, the new values will replace the current ones, provided the data type and count are not different. If either the data type or the count have been changed, Vsetattr will return FAIL (or -1).

The parameter data\_type is an integer number specifying the data type of the attribute values. Refer to Table 2F on page 14 for the definition of the data types to interpret this value. The parameter n\_values specifies the number of values to be stored in the attribute. The buffer attr\_values contains the values to be stored in the attribute.

Note that the FORTRAN-77 version of Vsetattr has two routines; vfsnatt sets a numeric value attribute and vfscatt sets a character value attribute.

Vsetattr returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5L.

### Retrieving the Index of a Vgroup Attribute Given the Attribute Name: Vfindattr

Vfindattr searches the vgroup, identified by the parameter vgroup\_id, for the attribute with the name specified by the parameter attr\_name, and returns the index of that attribute. The syntax of this routine is as follows:

C: attr\_index = Vfindattr(vgroup\_id, attr\_name);

FORTRAN: attr\_index = vffdatt(vgroup\_id, attr\_name)

Vfindattr returns either an attribute index or FAIL (or -1). The parameters of this routine are further defined in Table 5L.

* Vgetversion, Vsetattr, and Vfindattr Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vgetversion  [int32]  (vfgver) | vgroup\_id | int32 | integer | Vgroup identifier |
| Vsetattr  [intn]  (vfsnatt/vfscatt) | vgroup\_id | int32 | integer | Vgroup identifier |
| attr\_name | char \* | character\*(\*) | Name of the attribute |
| data\_type | int32 | integer | Data type of the attribute |
| n\_values | int32 | integer | Number of values the attribute contains |
| attr\_values | VOIDP | <valid numeric data type>(\*)/  character\* (\*) | Buffer containing the attribute values |
| Vfindattr  [intn]  (vffdatt) | vgroup\_id | int32 | integer | Vgroup identifier |
| attr\_name | char \* | character\*(\*) | Name of the target attribute |

### Obtaining the Total Number of Vgroup Attributes: Vnattrs and Vnattrs2

Both Vnattrs and Vnattrs2 return the number of attributes assigned to the vgroup specified by the parameter vgroup\_id, but Vnattrs2 is an updated version of Vnattrs. The syntax of both functions are as follows:

C: num\_of\_attrs = Vnattrs(vgroup\_id);

num\_of\_attrs = Vnattrs2(vgroup\_id);

FORTRAN: num\_of\_attrs = vfnatts(vgroup\_id)

Unvailable

There are two types of attributes for vgroups. One is the old-style that was created using methods other than the standard attribute API function Vsetattr, which was introduced after HDF Version 4.0 Release 2, July 19, 1996. Without the use of Vsetattr, an application could simulate an attribute for a vgroup by creating and writing a vdata of class \_HDF\_ATTRIBUTE and adding that vdata to the vgroup via these calls:

vdata\_ref = VHstoredatam(file\_id, ATTR\_FIELD\_NAME, values, size, type, attr\_name, \_HDF\_ATTRIBUTE, order);

ret\_value = Vaddtagref (vgroup\_id, DFTAG\_VH, vdata\_ref);

While both types of attributes are stored as vdatas, the vdatas of the two types of attributes are saved differently in the file. Because of the different storages, the new-style attribute functions, such as Vnattrs, Vgetattr or Vattrinfo, would miss the old-style attributes. Starting in release 4.2.6, new functions were added to allow applications to get access to both types of attributes, i.e., Vnattrs2, Vattrinfo2, and Vgetattr2.

Note that, when a vgroup has both type of attributes, the old-style attributes will preceed the new ones, regardless of when they were created. Applications that anticipate to access files that were created by HDF Version 4.0 Release 2 and before (circa July 1996,) should use Vnattrs2 instead of Vnattrs in order to include the old-style attributes if they exist and are desired.

Vnattrs and Vnattrs2 both returns the number of attributes, if successful, or FAIL (or -1), otherwise. These routines are further defined in Table 5M.

* Vnattrs and Vnattrs2 Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vnattrs  [intn]  (vfnatts) | vgroup\_id | int32 | integer | Vgroup identifier |
| Vnattrs2  [int32]  (Unavailable) | vgroup\_id | int32 | integer | Vgroup identifier |

### Obtaining Information on a Given Vgroup Attribute: Vattrinfo

Vattrinfo retrieves the name, data type, number of values, and the size of the values of an attribute that belongs to the vgroup identified by the parameter vgroup\_id. The syntax of Vattrinfo is as follows:

C: status = Vattrinfo(vgroup\_id, attr\_index, attr\_name, &data\_type, &n\_values, &size);

FORTRAN: status = vfainfo(vgroup\_id, attr\_index, attr\_name, data\_type, n\_values, size)

Vattrinfo stores the name, data type, number of values, and the size of the value of the attribute into the parameters attr\_name, data\_type, n\_values, and size, respectively.

The attribute is specified by its index, attr\_index. The valid values of attr\_index range from 0 to the total number of attributes attached to the vgroup - 1. The number of vgroup attributes can be obtained using the routine Vnattrs.

The parameter data\_type is an integer number. Refer to Table 2F on page 14 for the definitions of the data types to interpret this value. The parameter size contains the number of bytes taken by an attribute value.

In C, the parameters attr\_name, data\_type, n\_values, and size can be set to NULL, if the information returned by these parameters is not needed.

Note that, when working with HDF files that were created by HDF Version 4.0 Release 2 and before (circa July 1996,) please refer to the section about Vattrinfo2.

Vattrinfo returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further described in (See Table 5N on page 253).

### Obtaining Information on a Given Vgroup Attribute: Vattrinfo2

Vattrinfo2 is an updated version of Vattrinfo. Beside retrieving the name, datatype, number of values, and value size of an attribute identified by its index, *attr\_index,* in the vgroup, *vgroup\_id* as Vattrinfo, Vattrinfo2 also provides the reference number of and the number of fields in the vdata that represents the attribute.

The syntax of Vattrinfo2 is as follows:

C: status = Vattrinfo2(vgroup\_id, attr\_index, attr\_name, &data\_type, &n\_values, &size, &n\_fields, &ref\_num);

FORTRAN: Unavailable

The attribute is specified by its index, attr\_index. The valid values of attr\_index range from 0 to the total number of attributes attached to the vgroup - 1. The number of vgroup attributes can be obtained using the routine Vnattrs2.

The parameter data\_type is an integer number. Refer to Table 2F on page 14 for the definitions of the data types to interpret this value. The parameter size contains the number of bytes taken by an attribute value.

In C, the parameters attr\_name, data\_type, n\_values, and size can be set to NULL, if the information returned by these parameters is not needed.

Note that, this function should be used in place of Vattrinfo when working with HDF files that were created by HDF Version 4.0 Release 2 and before (circa July 1996.) Please refer to Section 5.8.4 on page 250 and the Appendix Attribute for more details about vgroup attributes.

Vattrinfo2 returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further described in Table 5N.

* Vattrinfo and Vattrinfo2 Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vattrinfo  [intn]  (vfainfo) | vgroup\_id | int32 | integer | Vgroup identifier |
| attr\_index | intn | integer | Index of the attribute |
| attr\_name | char \* | character\*(\*) | Returned name of the attribute |
| data\_type | int32 \* | integer | Returned data type of the attribute |
| n\_values | int32 \* | integer | Returned number of values of the attribute |
| size | int32 \* | integer | Returned size, in bytes, of the value of the attribute |
| Vattrinfo2  [intn]  (Unvailable) | vgroup\_id | int32 | N/A | Vgroup identifier |
| attr\_index | intn | N/A | Index of the attribute |
| attr\_name | char \* | N/A | Returned name of the attribute |
| data\_type | int32 \* | N/A | Returned data type of the attribute |
| n\_values | int32 \* | N/A | Returned number of values of the attribute |
| size | int32 \* | N/A | Returned size, in bytes, of the value of the attribute |
| n\_fields | int32 \* | N/A | Returned number of fields in the attribute vdata |
| ref\_num | uint16 \* | N/A | Returned reference number of the attribute vdata |

### Retrieving the Values of a Given Vgroup Attribute: Vgetattr

Vgetattr retrieves the values of an attribute of the vgroup specified by the parameter vgroup\_id. The syntax of Vgetattr is as follows:

C: status = Vgetattr(vgroup\_id, attr\_index, attr\_values);

FORTRAN: status = vfgnatt(vgroup\_id, attr\_index, attr\_values)

OR status = vfgcatt(vgroup\_id, attr\_index, attr\_values)

The attribute is specified by its index, attr\_index. The valid values of attr\_index range from 0 to the total number of attributes attached to the vgroup - 1. The number of vgroup attributes can be obtained using the routine Vnattrs.

The buffer attr\_values must be sufficiently allocated to hold the retrieved attribute values. Use Vattrinfo to obtain information about the attribute values for appropriate memory allocation.

Note that the FORTRAN-77 version of Vgetattr has two routines; vfgnatt gets a numeric value attribute and vfgcatt gets a character value attribute.

Vgetattr returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5O.

### Retrieving the Values of a Given Vgroup Attribute: Vgetattr2

As Vgetattr, Vgetattr2 retrieves the values of an attribute of the vgroup specified by the parameter vgroup\_id. The syntax of Vgetattr2 are as follows:

C: status = Vgetattr2(vgroup\_id, attr\_index, attr\_values);

FORTRAN: Currently unavailable

Unlike Vgetattr, Vgetattr2 can also read values from attributes that were created by methods other than Vsetattr. Please refer to Section 5.8.4 on page 250 and the Appendix Attribute for information about the different types of vgroup attributes.

The attribute is specified by its index, attr\_index. The valid values of attr\_index range from 0 to the total number of attributes attached to the vgroup - 1. The number of vgroup attributes can be obtained using the routine Vnattrs2.

The buffer attr\_values must be sufficiently allocated to hold the retrieved attribute values. Use Vattrinfo2 to obtain information about the attribute values for appropriate memory allocation.

Vgetattr2 returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routine are further defined in Table 5O.

* Vgetattr and Vgetattr2 Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vgetattr  [intn]  (vfgnatt/vfgcatt) | vgroup\_id | int32 | integer | Vgroup identifier |
| attr\_index | intn | integer | Index of the attribute |
| attr\_values | VOIDP | <valid numeric data type> (\*)/  character\*(\*) | Buffer containing attribute values |
| **Vgetattr2**  [intn]  (unavailable) | vgroup\_id | int32 | N/A | Vgroup identifier |
| attr\_index | intn | N/A | Index of the attribute |
| attr\_values | VOIDP | N/A | Buffer containing attribute values |

* Operations on Vgroup Attributes

This example illustrates the use of Vfind/vfind to locate a vgroup by its name, Vsetattr/vfscatt to attach an attribute to the vgroup, Vattrinfo/vfainfo to obtain information about the vgroup attribute, and Vgetattr/vfgcatt to obtain the attribute values.

The program obtains the version of the group then sets an attribute named "First Attribute" for the vgroup named "SD Vgroup". Next, the program gets the number of attributes that the vgroup has, and obtains and displays the name, the number of values, and the values of each attribute.

C version

C:

#include "hdf.h"

#define FILE\_NAME "General\_Vgroups.hdf"

#define VGROUP\_NAME "SD Vgroup"

#define VGATTR\_NAME "First Attribute"

#define N\_ATT\_VALUES 7 /\* number of values in the attribute \*/

main( )

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

intn status\_n, /\* returned status for functions returning an intn \*/

n\_attrs; /\* number of attributes of the vgroup \*/

int32 status\_32, /\* returned status for functions returning an int32 \*/

file\_id, vgroup\_ref, vgroup\_id,

attr\_index, i, vg\_version,

data\_type, n\_values, size;

char vg\_attr[N\_ATT\_VALUES] = {’v’,’g’,’r’,’o’,’u’,’p’,’\0’};

char vgattr\_buf[N\_ATT\_VALUES], attr\_name[30];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Open the HDF file for writing.

\*/

file\_id = Hopen (FILE\_NAME, DFACC\_WRITE, 0);

/\*

\* Initialize the V interface.

\*/

status\_n = Vstart (file\_id);

/\*

\* Get the reference number of the vgroup named VGROUP\_NAME.

\*/

vgroup\_ref = Vfind (file\_id, VGROUP\_NAME);

/\*

\* Attach to the vgroup found.

\*/

vgroup\_id = Vattach (file\_id, vgroup\_ref, "w");

/\*

\* Get and display the version of the attached vgroup.

\*/

vg\_version = Vgetversion (vgroup\_id);

switch (vg\_version) {

case VSET\_NEW\_VERSION:

printf ("\nVgroup %s is of the newest version, version 4\n",

VGROUP\_NAME);

break;

case VSET\_VERSION:

printf ("Vgroup %s is of a version between 3.2 and 4.0r2\n",

VGROUP\_NAME);

break;

case VSET\_OLD\_VERSION:

printf ("Vgroup %s is of version before 3.2\n", VGROUP\_NAME);

break;

default:

printf ("Unknown version = %d\n", vg\_version);

} /\* switch \*/

/\*

\* Add the attribute named VGATTR\_NAME to the vgroup.

\*/

status\_n = Vsetattr (vgroup\_id, VGATTR\_NAME, DFNT\_CHAR, N\_ATT\_VALUES,

vg\_attr);

/\*

\* Get and display the number of attributes attached to this vgroup.

\*/

n\_attrs = Vnattrs (vgroup\_id);

printf ("\nThis vgroup has %d attribute(s)\n", n\_attrs);

/\*

\* Get and display the name and the number of values of each attribute.

\* Note that the fourth and last parameters are set to NULL because the type

\* and the size of the attribute are not desired.

\*/

for (attr\_index = 0; attr\_index < n\_attrs; attr\_index++)

{

status\_n = Vattrinfo (vgroup\_id, attr\_index, attr\_name, NULL,

&n\_values, NULL);

printf ("\nAttribute #%d is named %s and has %d values: ",

attr\_index+1, attr\_name, n\_values);

/\*

\* Get and display the attribute values.

\*/

status\_n = Vgetattr (vgroup\_id, attr\_index, vgattr\_buf);

for (i = 0; i < n\_values; i++)

printf ("%c ", vgattr\_buf[i]);

printf ("\n");

}

/\*

\* Terminate access to the vgroup and to the V interface, and close

\* the HDF file.

\*/

status\_32 = Vdetach (vgroup\_id);

status\_n = Vend (file\_id);

status\_n = Hclose (file\_id);

}

FORTRAN-77 version

FORTRAN:

program vgroup\_attribute

implicit none

C

C Parameter declaration

C

character\*19 FILE\_NAME

character\*9 VGROUP\_NAME

character\*15 VGATTR\_NAME

C

parameter (FILE\_NAME = ’General\_Vgroups.hdf’,

+ VGROUP\_NAME = ’SD Vgroup’,

+ VGATTR\_NAME = ’First Attribute’)

integer VSET\_NEW\_VERSION, VSET\_VERSION, VSET\_OLD\_VERSION

parameter (VSET\_NEW\_VERSION = 4,

+ VSET\_VERSION = 3,

+ VSET\_OLD\_VERSION = 2)

integer DFACC\_WRITE

parameter (DFACC\_WRITE = 2)

integer DFNT\_CHAR

parameter (DFNT\_CHAR = 4)

integer N\_ATT\_VALUES

parameter (N\_ATT\_VALUES = 6)

C

C Function declaration

C

integer hopen, hclose

integer vfstart, vfatch, vfgver, vfscatt, vfnatts, vfainfo,

+ vfind, vfgcatt, vfdtch, vfend

C

C\*\*\*\* Variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

integer status, n\_attrs

integer file\_id

integer vgroup\_id, vgroup\_ref, vg\_version

integer attr\_index, i

integer data\_type, n\_values, size

character vg\_attr(N\_ATT\_VALUES)

character vgattr\_buf(N\_ATT\_VALUES), attr\_name(30)

data vg\_attr /’v’,’g’,’r’,’o’,’u’,’p’/

C

C\*\*\*\* End of variable declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C

C

C Open the HDF file for reading/writing.

C

file\_id = hopen(FILE\_NAME, DFACC\_WRITE, 0)

C

C Initialize the V interface.

C

status = vfstart(file\_id)

C

C Get the reference number of the vgroup named VGROUP\_NAME.

C

vgroup\_ref = vfind(file\_id, VGROUP\_NAME)

C

C Attach to the vgroup found.

C

vgroup\_id = vfatch(file\_id, vgroup\_ref , ’w’)

C

C Get and display the version of the attached vgroup.

C

vg\_version = vfgver(vgroup\_id)

if (vg\_version .eq. VSET\_NEW\_VERSION) write(\*,\*)

+ VGROUP\_NAME, ’ is of the newest version, version 4’

if (vg\_version .eq. VSET\_VERSION) write(\*,\*)

+ VGROUP\_NAME, ’ is of a version between 3.2 and 4.0r2’

if(vg\_version .eq. VSET\_OLD\_VERSION) write(\*,\*)

+ VGROUP\_NAME, ’ is of version before 3.2’

if ((vg\_version .ne. VSET\_NEW\_VERSION) .and.

+ (vg\_version .ne. VSET\_VERSION) .and.

+ (vg\_version .ne. VSET\_OLD\_VERSION)) write(\*,\*)

+ ’Unknown version’

C

C Add the attribute named VGATTR\_NAME to the vgroup.

C

status = vfscatt(vgroup\_id, VGATTR\_NAME, DFNT\_CHAR, N\_ATT\_VALUES,

+ vg\_attr)

C

C Get and display the number of attributes attached to this group.

C

n\_attrs = vfnatts(vgroup\_id)

write(\*,\*) ’This group has’, n\_attrs, ’ attributes’

C

C Get and display the name and the number of values of each attribute.

C

do 10 attr\_index=1, n\_attrs

status = vfainfo(vgroup\_id, attr\_index-1, attr\_name, data\_type,

+ n\_values, size)

write(\*,\*) ’Attribute #’, attr\_index-1, ’ is named ’, attr\_name

write(\*,\*) ’and has’, n\_values, ’ values: ’

C

C Get and display the attribute values.

C

status = vfgcatt(vgroup\_id, attr\_index-1, vgattr\_buf)

write(\*,\*) (vgattr\_buf(i), i=1,n\_values)

10 continue

C

C Terminate access to the vgroup.

C

status = vfdtch(vgroup\_id)

C

C Terminate accessto the V interface and close the HDF file.

C

status = vfend(file\_id)

status = hclose(file\_id)

end

## Obsolete Vgroup Interface Routines

The following routines have been replaced by newer routines with similar functionality. These routines are still supported by the Vgroup interface, but their use is not recommended. HDF may not support these routines in a future version.

### Determining the Next Vgroup or Vdata Identifier: Vgetnext

Vgetnext gets the reference number of the next member of a vgroup. This member can be either a vgroup or vdata. The syntax for Vgetnext is as follows:

C: ref\_num = Vgetnext(vgroup\_id, v\_ref);

FORTRAN: ref\_num = vfgnxt(vgroup\_id, v\_ref)

Vgetnext searches the vgroup, identified by the parameter vgroup\_id, for the vgroup or vdata whose reference number is specified by the parameter v\_ref. If this vgroup or vdata is found, Vgetnext finds the next vgroup or vdata and returns its reference number. If v\_ref is set to -1, the routine will return the reference number of the first vgroup or vdata in the vgroup.

Vgetnext is now obsolete as the routine Vgettagref provides the same functionality. In addition, Vgettagref is not restricted to searching for members that are vgroups or vdatas.

Vgetnext returns a reference number if the next vgroup or vdata is found, or FAIL (or -1) when an error occurs or when there are no more vdatas or vgroups in the vgroup. The parameters of Vgetnext are further defined in Table 5P.

### Determining the Number of Members and Vgroup Name: Vinquire

Vinquire retrieves the number of data objects and the name of the vgroup identified by the parameter vgroup\_id. The syntax for Vinquire is as follows:

C: status = Vinquire(vgroup\_id, &n\_members, vgroup\_name);

FORTRAN: status = vfinq(vgroup\_id, n\_members, vgroup\_name)

Vinquire stores the number of data objects and the vgroup name in the parameters n\_members and vgroup\_name, respectively. In C, if either n\_members or vgroup\_name is set to NULL, the corresponding data is not returned. The maximum length of the vgroup’s name is defined by VGNAMELENMAX (or 64).

Vinquire is now obsolete as the Vntagrefs routine can be used to get the number of data objects in a vgroup and Vgetname can be used to retrieve the name of a vgroup.

Vinquire returns either SUCCEED (or 0) or FAIL (or -1). The parameters of this routines are further defined in Table 5P.

* Vgetnext and Vinquire Parameter Lists

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vgetnext  [int32]  (vfgnxt) | vgroup\_id | int32 | integer | Vgroup identifier of the parent vgroup |
| v\_ref | int32 | integer | Reference number for the target vgroup |
| Vinquire  [intn]  (vfinq) | vgroup\_id | int32 | integer | Vgroup identifier |
| n\_members | int32 \* | integer | Pointer to the number of entries in the vgroup |
| vgroup\_name | char \* | character\*(\*) | Buffer for the name of the vgroup |