* Raw Data Information

# Chapter Overview

In 2011, to support the HDF4 File Content Map Project, HDF 4.2.6 introduced a set of routines that allow applications to access the raw data directly by providing the locations and sizes (i.e., offsets and lengths) of the data in an HDF file. The data can be all in one block or scattered in various locations due to linked-block or chunking storage scheme. This chapter describes these data information retrieval functions and provide examples of their usage.

## The Data Information Retrieval Routines

There are several of the data information retrieval functions across the AN, SD, GR, V, and VS interfaces and the prefix of each function’s name follows the same rule as other functions in the same interface. They all have "datainfo" in their names because their purpose is data information retrieval. Table 16A lists these routines. Currently, there is no implementation of the Fortran versions for these functions.

* Raw Data Information Retrieval Routines

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Routine Name | | Description and Reference |
| C | FORTRAN-77 |
| AN | ANgetdatainfo | unavailable | Retrieves data information of an annotation’s data (Section 16.3.1 on page 504) |
| SD | SDgetanndatainfo | unavailable | Retrieves data information of an DFSD API annotation’s data (Section 16.4.4 on page 508) |
| SDgetattdatainfo | unavailable | Retrieves offset and length of an SD API attribute’s data (Section 16.4.2 on page 506) |
| SDgetdatainfo | unavailable | Retrieves offset and length of a data set’s data (Section 16.4.1 on page 505) |
| SDgetoldattdatainfo | unavailable | Retrieves offset and length of a DFSD API attribute’s data (Section 16.4.3 on page 506) |
|
| GR | GRgetattdatainfo | unavailable | Retrieves offset and length of a GR API attribute’s data (Section 16.5.2 on page 510) |
| GRgetdatainfo | unavailable | Retrieves offset and length of a raster image’s data (Section 16.5.1 on page 509) |
| V | Vgetattdatainfo | unavailable | Retrieves offset and length of a V API attribute’s data (Section 16.6.1 on page 510) |
|
|
| VS | VSgetattdatainfo | unavailable | Retrieves offset and length of a VS API attribute’s data (Section 16.7.2 on page 512) |
| VSgetdatainfo | unavailable | Retrieves offset and length of a vdata or a vdata field’s data (Section 16.7.1 on page 511) |

There is no additional header file required for these new functions. As with existing API functions, the header file mfhdf.h must be included in programs that invoke SD interface routines, and hdf.h for non-SD ones.

## Addition to the AN Interface

There is one routine added to the AN API for raw data information retrieval, ANgetdatainfo, and it is described in the following sub-section.

### Retrieving Data Information of an Annotation: ANgetdatainfo

ANgetdatainfo retrieves the offset and length locating the data in a specified annotation. The syntax of ANgetdatainfo is as follows:

C: status = ANgetdatainfo(ann\_id, &offset, &length);

FORTRAN: Currently unavailable

The annotation is specified by its identifier, ann\_id. The offset and length are retrieved into the user-supplied buffers offset and length. Note that annotation’s data is stored in one contiguous block only.

ANgetdatainfo returns SUCCEED (or 0), if successful, or FAIL (or -1), otherwise. The parameters of ANgetdatainfo are specified in Table 16B.

* ANgetdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| ANgetdatainfo  [intn]  (unavailable) | ann\_id | int32 | N/A | Annotation identifier |
|
|
|
| offset | int32 \* | N/A | Buffer for offset of annotation’s data |
| length | int32 \* | N/A | Buffer for length of annotation’s data |

## Addition to the SD Interface

There are several functions added to the SD API for raw data information retrieval:

* SDgetdatainfo gets offsets/lengths of a data set’s data
* SDgetattdatainfo gets offset/length of SD API attribute’s data
* SDgetoldattdatainfo gets offset/length of DFSD API attribute’s data
* SDgetanndatainfo gets offset/length of an DFSD API annotation’s data

These functions are described in the following sub-sections.

### Retrieving Data Information of an SDS: SDgetdatainfo

SDgetdatainfo retrieves offset and length of data blocks in a specified data set. The syntax of SDgetdatainfo is as follows:

C: info\_count = SDgetdatainfo(sds\_id, origin, start\_block, info\_count, offsetarray, lengtharray);

FORTRAN: Currently unavailable

The offsets and lengths are retrieved into the user-supplied lists offsetarray and lengtharray.

* When the data set is contiguous, i.e., only one block of data, SDgetdatainfo will return a single pair of offset and length specifying the position of that data block.
* When the data set’s data is stored in linked-blocks, SDgetdatainfo will return a list of offsets and a list of lengths, each matching offset/length pair specifying the position of a linked block.
* When the data set has chunked data without linked-block storage, SDgetdatainfo will return a single pair of offset and length and, with linked-block storage, two list of offsets and lengths specifying the blocks in the chunk.

The parameter origin must be NULL when the data is not stored in chunking layout. When the data is chunked, SDgetdatainfo can be called on a single chunk and origin is used to specify the coordinates of the chunk.

The parameter info\_count specifies the maximum number of items the offset and length lists are allocated to hold. Applications, however, can pass in 0 for info\_count and NULL for these arrays when only the actual number of data blocks in the data set is desired.

The purpose of the parameter start\_block was to allow retrieval to start at a random block in the data. Applications would be able to start retrieving at the begining of the data by specifying start\_block as 0, or at a block of data by specifying start\_block as a value between 1 and the number of blocks in the data. However, in release 2.6, start\_block has no effect except for contiguous data, in which case, SDgetdatainfo will fail when start\_block is greater than 1. The supporting project did not need this specific feature. Thus, until the feature is supported, applications should pass 0 in for start\_block to start retrieving at the beginning of the data and up to info\_count or the total number of data blocks, whichever smaller.

SDgetdatainfo returns the number of offset/length pairs retrieved, if successful, or FAIL (or -1), otherwise. The parameters of SDgetdatainfo are specified in Table 16C.

* SDgetdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| SDgetdatainfo  [intn]  (unavailable) | sds\_id | int32 | N/A | Data set identifier |
| origin | int32 \* | N/A | Coordinates of the origin of the chunk to be read |
| start\_block | uintn | N/A | Indicating where to start reading offsets |
| info\_count | uintn | N/A | Length of the offset and length lists |
| offsetarray | int32 \* | N/A | Array to hold offsets of the data blocks |
| lengtharray | int32 \* | N/A | Array to hold lengths of the data blocks |

### Retrieving Data Information of an Attribute: SDgetattdatainfo

SDgetattdatainfo retrieves offset and length of the data in a specified attribute. The syntax of SDgetattdatainfo is as follows:

C: info\_count = SDgetattdatainfo(id, attr\_index, &offset, &length);

FORTRAN: Currently unavailable

The attribute is specified by its index and can be one that belongs to an SD file, a data set, or a dimension. The offset and length are retrieved into the user-supplied buffers offset and length. Note that attribute’s data is stored in one contiguous block only.

There are attributes created by SDsetattr and those created by the DFSD API functions. Refer to Appendix C, Attributes in HDF, for more details. SDgetattdatainfo can only retrieve data information of attributes that were created by SDsetattr. If the inquired attribute was created by the DFSD API functions, SDgetattdatainfo will return to the caller with error code DFE\_NOVGREP and the caller can call SDgetoldattdatainfo to get the attribute’s data information.

SDgetattdatainfo returns the number of offset/length pair retrieved, which should be 1, if successful, or FAIL (or -1), otherwise. The parameters of SDgetattdatainfo are specified in Table 16D.

* SDgetattdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| SDgetattdatainfo  [intn]  (unavailable) | id | int32 | N/A | SD, SDS, or dimension identifier |
| attr\_index | int32 | N/A | Index of the attribute being inquired |
|
|
| offset | int32 \* | N/A | Buffer for offset of attribute’s data |
| length | int32 \* | N/A | Buffer for length of attribute’s data |

### Retrieving Data Information of a DFSD API Attribute: SDgetoldattdatainfo

SDgetoldattdatainfo retrieves offset and length of the data in a specified attribute, which was created by the DFSD API routines. The attributes created in this manner were not stored as vdatas like those created by SDsetattr. These type of attributes are often seen in some older files, circa 1993. However, later files may still contain them if the file was written with the DFSD API routines. In addition, this type of attributes can only be predefined; there are no user-defined attributes in DFSD API.

SDgetoldattdatainfo only works on DFSD-created attributes while its counter part SDgetattdatainfo only works on attributes created with SDsetattr. An application might call SDgetattdatainfo initially. When a DFSD-created attribute is encountered, SDgetattdatainfo will fail with the error code DFE\_NOVGREP, which means there is no vgroup representation for the SDS and the SDS' attributes are stored differently than when they are created with SDsetattr. The application must call SDgetoldattdatainfo to get the data information of those attributes, if such error code is detected. For further information about this attribute issue, please refer to the Appendix C, Attributes in HDF, in this document. The syntax of SDgetoldattdatainfo is as follows:

C: info\_count = SDgetoldattdatainfo(dim\_id, sds\_id, attr\_name, &offset, &length);

FORTRAN: Currently unavailable

SDgetoldattdatainfo takes both SDS identifier and dimension identifier if the inquired attribute belongs to a dimension. When the inquired attribute belongs to an SDS, the dimension identifier will not be needed, and should be 0.

The attribute can be one that belongs to a data set or a dimension and is specified by its name, which can be one of the predefined names in (See Table 16E on page 508). The offset and length are retrieved into the user-supplied buffers offset and length. Note that attribute’s data is stored in one contiguous block only.

* HDF4 Predefined Attributes

|  |  |  |
| --- | --- | --- |
| Predefined Name | Actual Text | Applicable To |
|
| \_HDF\_LongName | "long\_name" | Dimension & SDS |
| \_HDF\_Units | "units" | Dimension & SDS |
| \_HDF\_Format | "format" | Dimension & SDS |
| \_HDF\_CoordSys | "coordsys" | Only SDS |
| \_HDF\_ScaleFactorErr | "scale\_factor\_err" | Only SDS |
| \_HDF\_AddOffset | "add\_offset" | Only SDS |
| \_HDF\_ValidRange | "valid\_range" | Only SDS |
| \_HDF\_ScaleFactor | "scale\_factor" | Only SDS |
| \_HDF\_AddOffsetErr | "add\_offset\_err" | Only SDS |
| \_HDF\_CalibratedNt | "calibrated\_nt" | Only SDS |
| \_HDF\_ValidMax | "valid\_max" | Only SDS |
| \_HDF\_ValidMin | "valid\_min" | Only SDS |
| \_FillValue | "\_FillValue" | Only SDS |

SDgetoldattdatainfo returns the number of offset/length pair retrieved, which should be 1, if successful, or FAIL (or -1), otherwise. The parameters of SDgetoldattdatainfo are specified in Table 16F.

* SDgetoldattdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| SDgetoldattdatainfo  [intn]  (unavailable) | dim\_id | int32 | N/A | Dimension identifier |
| sds\_id | int32 | N/A | SDS identifier |
| attr\_name | char \* | N/A | Name of the attribute being inquired |
| offset | int32 \* | N/A | Buffer for offset of attribute’s data |
|
| length | int32 \* | N/A | Buffer for length of attribute’s data |

### Retrieving Data Information of an Annotation in SD API: SDgetanndatainfo

SDgetanndatainfo retrieves offsets and lengths of the data belonging to the annotations of a given type. These annotations were created with the DFAN API. The syntax of SDgetanndatainfo is as follows:

C: info\_count = SDgetanndatainfo(id, annotype, size, offsetarray, lengtharray);

FORTRAN: Currently unavailable

The parameter id can be an SD or SDS identifier. However, when id is an SD identifier, the annotation’s type must be either AN\_FILE\_LABEL (or 2) or AN\_FILE\_DESC (or 3), and when it is an SDS identifier, the type must be AN\_DATA\_LABEL (or 0) or AN\_DATA\_DESC (or 1). The offsets and lengths of the specified annotations are retrieved into the user-supplied buffers offsetarray and lengtharray. Note that annotation’s data is stored in one contiguous block only, but there can be more than one annotation of the specified type. The parameter size specifies the number of elements offsetarray and lengtharray can hold.

SDgetanndatainfo returns the number of offset/length pairs retrieved, if successful, or FAIL (or -1), otherwise. The parameters of SDgetanndatainfo are specified in Table 16G.

* SDgetanndatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| SDgetanndatainfo  [intn]  (unavailable) | id | int32 | N/A | SD or SDS identifier |
| annotype | ann\_type | N/A | Type of annotations to retrieve data info |
| size | uintn | N/A | Length of the offset and length arrays |
|
| offsetarray | int32 \* | N/A | Buffer for offset of annotations’ data |
| lengtharray | int32 \* | N/A | Buffer for length of annotations’ data |

## Addition to the GR Interface

There are two routines added to the GR API for raw data information retrieval, GRgetdatainfo and GRgetattdatainfo, and they are described in the following sub-sections.

### Retrieving Data Information of a Raster Image: GRgetdatainfo

GRgetdatainfo retrieves offset and length of data blocks in a specified raster image. The syntax of GRgetdatainfo is as follows:

C: info\_count = GRgetdatainfo(ri\_id, start\_block, info\_count, offsetarray, lengtharray);

FORTRAN: Currently unavailable

The offsets and lengths are retrieved into the user-supplied lists offsetarray and lengtharray.

* When the raster image is contiguous, i.e., only one block of data, GRgetdatainfo will return a single pair of offset and length specifying the position of that data block.
* When the raster image’s data is stored in linked-blocks, GRgetdatainfo will return a list of offsets and a list of lengths, each matching offset/length pair specifying the position of a linked block.
* GRgetdatainfo does not work with chunked images. (The HDF4 File Content Map Project did not need this feature.)

The parameter info\_count specifies the maximum number of items the offset and length lists are allocated to hold. Applications, however, can pass in 0 for info\_count and NULL for these arrays when only the actual number of data blocks in the data set is desired.

The purpose of the parameter start\_block was to allow retrieval to start at a random block in the data. Applications would be able to start retrieving at the begining of the data by specifying start\_block as 0, or at a block of data by specifying start\_block as a value between 1 and the number of blocks in the data. However, in release 2.6, start\_block has no effect except for contiguous data, in which case, GRgetdatainfo will fail when start\_block is greater than 1. The supporting project did not need this specific feature. Thus, until the feature is supported, applications should pass 0 in for start\_block to start retrieving at the beginning of the data and up to info\_count or the total number of data blocks, whichever smaller.

GRgetdatainfo returns the number of offset/length pairs retrieved, if successful, or FAIL (or -1), otherwise. The parameters of GRgetdatainfo are specified in Table 16H.

* GRgetdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| GRgetdatainfo  [intn]  (unavailable) | ri\_id | int32 | N/A | Raster image identifier |
|
| start\_block | uintn | N/A | Indicating where to start reading offsets |
| info\_count | uintn | N/A | Length of the offset and length lists |
| offsetarray | int32 \* | N/A | Array to hold offsets of the data blocks |
| lengtharray | int32 \* | N/A | Array to hold lengths of the data blocks |

### Retrieving Data Information of a GR API Attribute: GRgetattdatainfo

GRgetattdatainfo retrieves offset and length of the data in a specified attribute. The syntax of GRgetattdatainfo is as follows:

C: info\_count = GRgetattdatainfo(id, attr\_index, &offset, &length);

FORTRAN: Currently unavailable

The attribute is specified by its index and can be one that belongs to a GR file or a raster image. The offset and length are retrieved into the user-supplied buffers offset and length. Note that attribute’s data is stored in one contiguous block only.

GRgetattdatainfo returns the number of offset/length pair retrieved, which should be 1, if successful, or FAIL (or -1), otherwise. The parameters of GRgetattdatainfo are specified in Table 16I.

* GRgetattdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| GRgetattdatainfo  [intn]  (unavailable) | id | int32 | N/A | GR or raster image identifier |
| attr\_index | int32 | N/A | Index of the attribute being inquired |
|
|
| offset | int32 \* | N/A | Buffer for offset of attribute’s data |
| length | int32 \* | N/A | Buffer for length of attribute’s data |

## Addition to the V Interface

There is one routine added to the V API for raw data information retrieval, Vgetattdatainfo, and it is described in the following sub-section.

### Retrieving Data Information of a V API Attribute: Vgetattdatainfo

Vgetattdatainfo retrieves the offset and length locating the data in a specified attribute. The syntax of Vgetattdatainfo is as follows:

C: status = Vgetattdatainfo(vgroup\_id, attr\_index, &offset, &length);

FORTRAN: Currently unavailable

The annotation is specified by its identifier, ann\_id. The offset and length are retrieved into the user-supplied buffers offset and length. Note that annotation’s data is stored in one contiguous block only.

There are two types of attributes for vgroups; those created by Vsetattr (new style) and those created by non-Vsetattr approaches (old style.) Please refer to the section about Vnattrs and Vnattrs2 and the Appendix Attribute in this *HDF User’s Guide* for details. Vgetattdatainfo can access both type of attributes. However, an application must use Vnattrs2 to get the number of attributes instead of Vnattrs in order to include both types. Note that, when a vgroup has both types of attributes, the old-style attributes will preceed the new ones, regardless of when they were created. The best way to access these attributes is through a loop.

Vgetattdatainfo returns the number of data blocks, which should be 1, if successful, or FAIL (or -1), otherwise. The parameters of Vgetattdatainfo are specified in Table 16J.

* Vgetattdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| Vgetattdatainfo  [intn]  (unavailable) | vgroup\_id | int32 | N/A | Annotation identifier |
|
|
|
| attr\_index | intn | N/A | Index of the inquired attribute |
| offset | int32 \* | N/A | Buffer for offset of attribute’s data |
| length | int32 \* | N/A | Buffer for length of attribute’s data |

## Addition to the VS Interface

There are two routines added to the VS API for raw data information retrieval, VSgetdatainfo and VSgetattdatainfo, and they are described in the following sub-sections.

### Retrieving Data Information of a Vdata: VSgetdatainfo

VSgetdatainfo retrieves offset and length of data blocks in a specified vdata. The syntax of VSgetdatainfo is as follows:

C: info\_count = VSgetdatainfo(vdata\_id, start\_block, info\_count, offsetarray, lengtharray);

FORTRAN: Currently unavailable

The offsets and lengths are retrieved into the user-supplied lists offsetarray and lengtharray.

* When the vdata has is contiguous data, i.e., only one block of data, VSgetdatainfo will return a single pair of offset and length specifying the position of that data block.
* When the vdata’s data is stored in linked-blocks, VSgetdatainfo will return a list of offsets and a list of lengths, each matching offset/length pair specifying the position of a linked block.

The parameter info\_count specifies the maximum number of items the offset and length lists are allocated to hold. Applications, however, can pass in 0 for info\_count and NULL for these arrays when only the actual number of data blocks in the data set is desired.

The purpose of the parameter start\_block was to allow retrieval to start at a random block in the data. Applications would be able to start retrieving at the begining of the data by specifying start\_block as 0, or at a block of data by specifying start\_block as a value between 1 and the number of blocks in the data. However, in release 2.6, start\_block has no effect except for contiguous data, in which case, VSgetdatainfo will fail when start\_block is greater than 1. The supporting project did not need this specific feature. Thus, until the feature is supported, applications should pass 0 in for start\_block to start retrieving at the beginning of the data and up to info\_count or the total number of data blocks, whichever smaller.

VSgetdatainfo returns a the number of offset/length pairs retrieved, if successful, or FAIL (or -1), otherwise. The parameters of VSgetdatainfo are specified in Table 16K.

* VSgetdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| VSgetdatainfo  [intn]  (unavailable) | vdata\_id | int32 | N/A | Vdata identifier |
|
| start\_block | uintn | N/A | Indicating where to start reading offsets |
| info\_count | uintn | N/A | Length of the offset and length lists |
| offsetarray | int32 \* | N/A | Array to hold offsets of the data blocks |
| lengtharray | int32 \* | N/A | Array to hold lengths of the data blocks |

### Retrieving Data Information of a VS API Attribute: VSgetattdatainfo

VSgetattdatainfo retrieves offset and length of the data in a specified attribute. The syntax of VSgetattdatainfo is as follows:

C: info\_count = VSgetattdatainfo(vdata\_id, findex, attr\_index, &offset, &length);

FORTRAN: Currently unavailable

The attribute is specified by its index, attr\_index, and can be one that belongs to a vdata or a field of the vdata. If findex is \_HDF\_VDATA (or -1), then the attribute is associated with the vdata. If findex is an index of the vdata field, then the attribute is one that is associated with the vdata field. The parameter attr\_index specifies the attribute's index within the vdata's or the field's attribute list. Thus, its value must be within [0-number of attributes of the associated list].

The offset and length are retrieved into the user-supplied buffers offset and length. Note that attribute’s data is stored in one contiguous block only.

VSgetattdatainfo returns the number of offset/length pair retrieved, which should be 1, if successful, or FAIL (or -1), otherwise. The parameters of VSgetattdatainfo are specified in Table 16L.

* VSgetattdatainfo Parameter List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Routine Name  [Return Type]  (FORTRAN-77) | Parameter | Parameter Type | | Description |
| C | FORTRAN-77 |
| VSgetattdatainfo  [intn]  (unavailable) | vdata\_id | int32 | N/A | Vdata identifier |
| findex | int32 | N/A | Vdata's field index or \_HDF\_VDATA |
| attr\_index | int32 | N/A | Index of the attribute being inquired |
|
|
| offset | int32 \* | N/A | Buffer for offset of attribute’s data |
| length | int32 \* | N/A | Buffer for length of attribute’s data |

* Getting Data Information of SDS.

This example demonstrates the use of the routine SDgetdatainfo with simple and contiguous data in a data set.

C version

C:

#include "mfhdf.h"

#define SIMPLE\_FILE "datainfo\_simple.hdf" /\* data file previously written \*/

main()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Variable Declaration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int32 sd\_id, sds\_id;

int32 offset, length;

uintn info\_count = 0;

intn status;

/\*

\* Open the file for reading.

\*/

sd\_id = SDstart(SIMPLE\_FILE, DFACC\_READ);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Read data info for later accessing data without the use of HDF4 library

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Open the second dataset, get the number of data block, which is 1, then

\* retrieve and record the offset/length

\*/

sds\_id = SDselect(sd\_id, 1);

/\*

\* Passing in 0 for the info count and NULL for the offset and length

\* arrays to get the number of data blocks in the data set. Note that

\* the second parameter is for chunk coordinates and because this data

\* set is not chunked, NULL should be passed in. The third parameter

\* indicates to start retrieval at the beginning of the data.

\*/

info\_count = SDgetdatainfo(sds\_id, NULL, 0, 0, NULL, NULL);

/\*

\* Call SDgetdatainfo again to retrieve the offset and length of the

\* data block. The info count is now 1 to specify the number of elements

\* in the offset and length arrays.

\*/

status = SDgetdatainfo(sds\_id, NULL, 0, info\_count, &offset, &length);

/\*

\* Terminate access to the data set.

\*/

status = SDendaccess(sds\_id);

/\*

\* Close the file.

\*/

status = SDend(sd\_id);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Read data using previously obtained data info without HDF4 library

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Open file and read in data without using SD API \*/

{

int fd; /\* for open \*/

int32 ret32; /\* for DFKconvert \*/

ssize\_t readlen = 0; /\* for read \*/

int32 \*readibuf, \*readibuf\_swapped;

/\*

\* Open the file for reading without SD API.

\*/

fd = open(SIMPLE\_FILE, O\_RDONLY);

/\*

\* Forward to the position of the data.

\*/

lseek(fd, (off\_t)offset, SEEK\_SET);

/\*

\* Allocate buffers for SDS' data.

\*/

readibuf = (int32 \*) HDmalloc(N\_VALUES \* sizeof(int32));

readibuf\_swapped = (int32 \*) HDmalloc(N\_VALUES \* sizeof(int32));

/\*

\* Read in this block of data.

\*/

readlen = read(fd, (VOIDP) readibuf, (size\_t)length);

/\*

\* Convert data back to format on local machine.

\*/

ret32 = DFKconvert(readibuf, readibuf\_swapped, DFNT\_INT32,

N\_VALUES, DFACC\_WRITE, 0, 0);

/\*

\* Free resources.

\*/

HDfree (readibuf\_swapped);

HDfree (readibuf);

/\*

\* Close the file.

\*/

close(fd);

}

}