Hclose/hclose

intn Hclose(int32 *file\_id*)

|  |  |  |
| --- | --- | --- |
| file\_id | IN: | File identifier returned by Hopen |
| Purpose | Closes the access path to the file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The file identifier file\_id is validated before the file is closed. If the identifier is valid, the function closes the access path to the file. | | |
|  | If there are still access identifiers attached to the file, the error DFE\_OPENAID is placed on the error stack, FAIL (or -1) is returned, and the file remains open. This is a common error when developing new interfaces. Refer to the Reference Manual page on Hendaccess for a discussion of this problem. | | |
| FORTRAN | integer function hclose(file\_id) | | |
|  | integer file\_id | | |

Hgetfileversion/hgfilver

intn Hgetfileversion(int32 *file\_id*, uint32 \**major\_v*, uint32 \**minor\_v*, uint32 \**release*, char *string*[])

|  |  |  |
| --- | --- | --- |
| file\_id | IN: | File identifier returned by Hopen |
| major\_v | OUT: | Major version number |
| minor\_v | OUT: | Minor version number |
| release | OUT: | Release number |
| string | OUT: | Version number text string |
| Purpose | Retrieves version information for an HDF file. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | It is still an open question as to what exactly the version number of a file should mean, so we recommend that code not depend on this buffer. The string argument is limited to a length of LIBVSTR\_LEN (or 80) characters as defined in hfile.h. | | |
| FORTRAN | integer function hgfilver(file\_id, major\_v, minor\_v, release, string) | | |
|  | integer file\_id, major\_v, minor\_v, release | | |
|  | character\*(\*) string | | |

Hgetlibversion/hglibver

intn Hgetlibversion(uint32 \**major\_v*, uint32 \**minor\_v*, uint32 \**release*, char *string*[])

|  |  |  |
| --- | --- | --- |
| major\_v | OUT: | Major version number |
| minor\_v | OUT: | Minor version number |
| release | OUT: | Release number |
| string | OUT: | Version number text string |

|  |  |
| --- | --- |
| Purpose | Retrieves the version information of the current HDF library. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| Description | The version information is compiled into the HDF library, so it is not necessary to have any open files for this function to execute. The string buffer is limited to a length of LIBVSTR\_LEN (or 80) characters as defined in hfile.h. |

|  |  |
| --- | --- |
| FORTRAN | integer function hglibver(major\_v, minor\_v, release, string) |
|  | integer major\_v, minor\_v, release |
|  | character\*(\*) string |

Hgetntinfo

intn Hgetntinfo(const int32 *n\_type*, hdf\_ntinfo\_t \**nt\_info*)

|  |  |  |
| --- | --- | --- |
| n\_type | IN: | HDF4 number type |
| nt\_info | OUT: | Number type's information |
| Purpose | Retrieves some information of the given number type. | | |
| Return value | Returns SUCCEED (0) if successful, and FAIL (-1) otherwise. | | |
| Description | Hgetntinfo retrieves the name and byte order of the given number type, n\_type. These values are in character arrays and are encapsulated in the structure hdf\_ntinfo\_t, which is defined in hdf.h as:  typedef struct hdf\_ntinfo\_t  {  char type\_name[9]; /\* nt name, e.g. "int8" or "float32" \*/  char byte\_order[13]; /\* nt byte order, e.g. "littleEndian" or "bigEndian" \*/  }  hdf\_ntinfo\_t;  Hgetntinfo returns FAIL (-1) when n\_type does not match one of the types listed in the tables in Section 2.5.2, "HDF Definitions" of the HDF User's Guide. | | |
| FORTRAN | Currently unavailable | | |
|  |  | | |

Hishdf/hishdff

intn Hishdf(char \**filename*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Complete path and filename of the file to be checked. |
| Purpose | Determines if a file is an HDF file. | | |
| Return value | Returns TRUE (or 1) if the file is an HDF file and FALSE (or 0) otherwise. | | |
| Description | The first four bytes of a file identify it as an HDF file. It is possible that Hishdf will identify a file as an HDF file but Hopen will be unable to open the file; for example, if the data descriptor list is corrupted. | | |
| FORTRAN | integer function hishdff(filename) | | |
|  | character\*(\*) filename | | |

Hopen/hopen

int32 Hopen(char \**filename*, intn *access*, int16 *n\_dds*)

|  |  |  |
| --- | --- | --- |
| filename | IN: | Complete path and filename for the file to be opened |
| access | IN: | Access code definition (preceded by DFACC\_) |
| n\_dds | IN: | Number of data descriptors in a block if a new file is to be created |
| Purpose | Provides an access path to an HDF file by reading all the data descriptor blocks into memory. | | |
| Return value | Returns the file identifier if successful and FAIL (or -1) otherwise. | | |
| Description | If given a new file name, Hopen will create a new file using the specified access type and number of data descriptors. If given an existing file name, Hopen will open the file using the specified access type and ignore the n\_dds argument. | | |
|  | The number of data descriptors in a block, n\_dds, is a non-negative integer with a default value of DEF\_NDDS (or 16) and a minimum value of MIN\_NDDS (or 4). If the specified value of n\_dds is less than MIN\_NDDS, then it will be set to MIN\_NDDS. | | |
|  | HDF provides several access code definitions: | | |
|  | DFACC\_CREATE - Create a new file. If file exists, replace its contents.  DFACC\_READ - Open for read only. If file does not exist, return an error.  DFACC\_WRITE - Open for read/write. If file does not exist, create it. | | |
|  | If a file is opened and an attempt is made to reopen the file using DFACC\_CREATE, HDF will issue the error code DFE\_ALROPEN. If the file is opened with read-only access and an attempt is made to reopen the file for write access using DFACC\_WRITE, HDF will attempt to reopen the file with read and write permissions. | | |
|  | Upon successful exit, the specified file is opened with the relevant permissions, the data descriptors are set up in memory, and the associated file\_id is returned. For new files, the appropriate file headers are also set up. | | |
|  | Note that it has been reported that opening/closing file in loops is very slow; thus, it is not recommended to perform such operations too many times, particularly, when data is being added to the file between opening/closing. | | |
| FORTRAN | integer function hopen(filename, access, n\_dds) | | |
|  | character\*(\*) filename | | |
|  | integer access, n\_dds | | |

HCget\_config\_info

intn HCget\_config\_info(comp\_coder\_t *coder\_type*, uint32 \**compression\_config\_info*)

|  |  |  |  |
| --- | --- | --- | --- |
| coder\_type | | IN: | Type of compression |
| compression\_config\_info | | OUT: | Flags indicating status of compression method |
| Purpose | Retrieves information about the configuration of a compression method. | | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | | |
| Description | HCget\_config\_info retrieves the configuration status of the compression type specified by coder\_type, returning that status information as flags in compression\_config\_info. | | | |
|  | Valid values of coder\_type are as follows: | | | |
|  | COMP\_CODE\_NONE - for no compression  COMP\_CODE\_RLE - for RLE run-length encoding  COMP\_CODE\_NBIT - for NBIT compression  COMP\_CODE\_SKPHUFF - for Skipping Huffman compression  COMP\_CODE\_DEFLATE - for GZIP compression  COMP\_CODE\_SZIP - for Szip compression  COMP\_CODE\_JPEG - for JPEG compression  The compression method, coder\_type, used for a data set can be obtained as the returned value of the comp\_type parameter in an SDgetcompinfo call. | | | |
|  | The configuration flags returned in compression\_config\_info include the following:  0 - Compression method is not enabled.  COMP\_DECODER\_ENABLED - Decoding is enabled.  COMP\_ENCODER\_ENABLED - Encoding is enabled.  If the returned value is COMP\_DECODER\_ENABLED|COMP\_ENCODER\_ENABLED, the compression method is enabled for both encoding and decoding. | | | |
|  | In the general case, any available compression type can be configured in any mode:  COMP\_DECODER\_ENABLED  COMP\_ENCODER\_ENABLED  COMP\_DECODER\_ENABLED |COMP\_ENCODER\_ENABLED  As of this writing (HDF4 Release 2.1, February 2005), only the Szip compression library is actually used with the HDF libraries in more than one configuration (see immediately below.) As a third-party product, it is distributed in both decode-only and encode/decode configurations. All other compression methods are currently distributed or used in an encode/decode configuration if they are available at all, and HCget\_config\_info returns either 0 or COMP\_DECODER\_ENABLED|COMP\_ENCODER\_ENABLED when they are used. | | | |
|  | Due to licensing requirements, the Szip library is available in both decode-only and encode/decode configurations. Therefore, the full range of values can be returned for Szip compression.  **o** If the Szip version available on a system is decode-only, HCget\_config\_info will return COMP\_DECODER\_ENABLED in compression\_config\_info.  **o** If the available Szip library is configured as encode/decode, compression\_config\_info will contain the value COMP\_DECODER\_ENABLED|COMP\_ENCODER\_ENABLED upon return. | | | |
| Note | **Regarding Szip compression in HDF4:**  Szip compression is available only through the SD interface and is documented in the SDsetcompress and SDgetcompinfo reference manual entries. Aside from the configuration discovery capability documented in HCget\_config\_info, Szip compression is not accessible through the HC interface. | | | |
| See also | **Regarding Szip usage and licensing:**  See http://www.hdfgroup.org/doc\_resource/SZIP/ for information regarding the use of Szip in HDF products and Szip licensing. | | | |
|  | **Regarding compression in HDF4:**  See the SDsetcompress and SDgetcompinfo entries in this reference manual for a more general description of dataset compression information. | | | |
| FORTRAN | currently unavailable | | | |
|  |  | | | |

HDdont\_atexit/hddontatexit

intn HDdont\_atexit(void)

|  |  |
| --- | --- |
| Purpose | Indicates to the library that an atexit() routine is **\_not\_** to be installed. |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. |
| Description | This routine indicates to the library that an atexit() cleanup routine should not be installed. The purpose for this is in situations where the library is dynamically linked into an application and is unlinked from the application before exit() gets called. In those situations, a routine installed with atexit() would jump to a routine which was no longer in memory, causing errors. |
|  | In order to be effective, this routine must be called before any other HDF function calls, and must be called each time the library is loaded/linked into the application (the first time and after it has been unloaded). |
|  | If this routine is used, certain memory buffers will not be deallocated, although in theory a user could call HPend on their own. |
| FORTRAN | integer hddontatexit( ) |

HXsetcreatedir/hxiscdir

intn HXsetcreatedir(char \**dir*)

|  |  |  |
| --- | --- | --- |
| dir | IN: | Target directory of the external file to be written |
| Purpose | Initializes the directory environment variable, identifying the location of the external file to be written. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | The contents of dir is copied into the private memory of the HDF library. If dir is NULL, the directory variable is unset. If HXsetcreatedir encounters an error condition, the directory variable is not changed. When a new external element is created (via the routines HXcreate or SDsetexternal), the HDF library accesses the external file just like the open call by default. Refer to the Reference Manual page on HXcreate for a description of when a new or an old file should be opened. | | |
|  | Users may override the default action by calling HXsetcreatedir or by defining the environment variable $HDFEXTCREATEDIR. The HDF library will access the external file in the directory according to the environment variable setting. The precedence is HXsetcreatedir, then $HDXEXTDIR, in the manner of open. | | |
|  | Note that the above override does not apply to absolute pathnames - i.e., filenames starting with a forward slash. HDF will access the absolute pathname without change. Also note that HXsetcreatedir and $HDFEXTCREATEDIR are not symmetrical to HXsetdir and $HDFEXTDIR. The former pair permits only single directory values and is used to compose the filename for access. The later pair permits multiple directory values which are used for searching an existing file. | | |
|  | The dir\_len parameter in the FORTRAN-77 routine specifies the length of the dir character string. | | |
| FORTRAN | integer function hxiscdir(dir, dir\_len) | | |
|  | character\*(\*) dir | | |
|  | integer dir\_len | | |

HXsetdir/hxisdir

intn HXsetdir(char \**dir*)

|  |  |  |
| --- | --- | --- |
| dir | IN: | Target directory of the external file to be located |
| Purpose | Initializes the directory environment variable, identifying the location of the external file to be located. | | |
| Return value | Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise. | | |
| Description | HXsetdir sets the directory variable for locating an external file according to dir which may contain multiple directories separated by vertical bars (e.g., “dir1|dir2”). The content of dir is copied into the private memory of the HDF library. If dir is NULL, the directory variable is unset. | | |
|  | If HXsetdir encounters any error, the directory variable is not changed. By default, the HDF library locates the external file just like the open call. It also searches for the external file in the directories specified by the user environment variable $HDFEXTDIR, if defined, and the directory variable set by HXsetdir. The searching precedence is directory variable, if set, then $HDXEXTDIR, then in the manner of open. | | |
|  | The searching differs if the external filename is an absolute pathname - i.e., starting with a forward slash. HDF will try open first. If open fails and if $HDFEXTDIR is defined or the directory variable is set via HXsetdir, HDF will remove all directory components of the absolute pathname (e.g., “/usr/groupA/projectB/Data001” becomes “Data001”) and search for that filename with the strategy described in the previous paragraph. | | |
|  | The dir\_len parameter in the FORTRAN-77 routine specifies the length of the dir character string. | | |
| FORTRAN | integer function hxisdir(dir, dir\_len) | | |
|  | character\*(\*) dir | | |
|  | integer dir\_len | | |