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
```

Hishdf

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 corrupt.

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` - If file exists, delete it, then open a new file for read/write.
`DFACC_READ` - Open for read only. If file does not exist, 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_RDWR` or `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.

FORTRAN

```
integer function hopen(filename, access, n_dds)


character*(*) filename

integer access, n_dds
```

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( )`

HEprint/heprnt

VOID HEprint(FILE **stream*, int32 *level*)

stream	IN:	Stream to print error message to
level	IN:	Level of error stack to print

Purpose Prints information to the error stack.

Return value None.

Description If *level* is `0`, all of the errors currently on the error stack are printed. Output from this function is sent to the file pointed to by *stream.*

The following information is printed: the ASCII description of the error, the reporting routine, the reporting routine as source file name, and the line at which the error was reported. If the programmer has supplied extra information by means of **HEreport**, this information is printed as well.

The FORTRAN-77 routine uses one less parameter than the C routine because it doesn't allow the user to specify the print stream. Instead, it always prints to `stdout`.

FORTRAN

```
integer heprnt(level)


integer level
```

HEstring

char *HEstring(int16 *error_code*)

error_code	IN:	HDF error code

Purpose Returns the error message associated with specified error code.

Return value Returns a pointer to a string associated with the error code if successful.

Description Returns a text description of the given error code. These strings are statically declared and should not be deallocated from memory (using the `free` routine) by the user. If a defined text description cannot be found a generic default message is returned.

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
```