

HDc2fstr

HDc2fstr

intn HDc2fstr(char **string*, intn *length*)

string IN: Null-terminated C string to be converted

length IN: Length of the Fortran-77 string

Purpose Converts a C string to a Fortran-77 string in place.

Return value SUCCEED.

Description The null termination is deleted and the string is padded with spaces.

HDcalloc

VOIDP HDcalloc(uint32 *num_blks*, uint32 *blk_size*)

num_blks IN: Number of memory blocks to reallocate

blk_size IN: Size of each memory block

Purpose Dynamically reallocates a block of memory and returns a pointer to it after initializing the block locations to zeroes.

Return value A pointer to the allocated memory block if successful, `NULL` otherwise.

Description Operates in a manner similar to **calloc**. **HDcalloc** calls **HDmalloc** and **HDmemset**, saving the calling routine the necessity of doing this.

HDerr

HDerr

int HDerr(int32 *file_id*)

file_id IN: File identifier

Purpose Closes the specified file and returns `FAIL` (or `-1`).

Return value `FAIL` (or `-1`).

Description **HDerr** is a replacement for **DFIerr** in HDF version 3.1 and earlier versions.

Hdf2cstring

intn Hdf2cstring(*_fcd fdesc*, intn *length*)

fdesc IN: Fortran-77 string descriptor

length IN: Length of the Fortran-77 string

Purpose Converts a Fortran-77 string to a C string.

Return value Returns a pointer to the C string if successful and `NULL` otherwise.

Description **Hdf2cstring** deletes the trailing spaces in the Fortran-77 string. The user must deallocate the memory space of this string.

HDfidtoname

HDfidtoname

const char *HDfidtoname(int32 *file_id*)

file_id IN: File identifier

Purpose Returns the file name the specified file identifier corresponds to.

Return value Returns a pointer to the file name if successful or `NULL` otherwise.

Description **HDfidtoname** useful for mixing old-style single-file interfaces - which take filenames - and newer interfaces which use file identifiers.

HDgetc

intn HDgetc(int32 *h_id*)

h_id IN: Data element identifier

Purpose	Reads one byte from the specified data element.
Return value	Returns a byte read from the data element if successful and <code>FAIL</code> (or <code>-1</code>) otherwise
Description	HDgetc calls Hread to read the byte and reports any error codes returned by Hread .

HDgetNTdesc

HDgetNTdesc

char *HDgetNTdesc(int32 *nt*)

nt IN: Data type to be queried

Purpose .Returns a text description of a data type.

Return value .Returns a pointer to the description text if successful and `NULL` otherwise.

HDgettagdesc

const char *HDgettagdesc(uint16 *tag*)

tag IN: Tag of the element of be queried

Purpose Returns the text description of the specified tag.

Return value Returns a pointer to the description text if successful and `NULL` otherwise.

HDgettagnum

HDgettagnum

intn HDgettagnum(const char **tag_name*)

tag_name IN: Name of the tag to be queried

Purpose Returns the tag number corresponding to the text description of a tag.

Return value Returns the tag number, which is greater than or equal to 0, if successful or
FAIL (or -1) otherwise.

HDgettagsname

char *HDgettagsname(uint16 *tag*)

tag IN: Tag of the data element to be queried

- Purpose** Returns the name of the specified tag.
- Return value** Returns a pointer to the name if successful or `NULL` otherwise.
- Description** Also checks for special elements.

HDinqblockinfo

HDinqblockinfo

int HDinqblockinfo(int32 *h_id*, int32 **length*, int32 **first_length*, int32 **block_length*, int32
n_blocks)

h_id IN: Access identifier of the linked-block element

length OUT: Length of the element

first_length OUT: Length of the first block

block_length OUT: Standard length of the blocks

n_blocks OUT: Number of blocks

Purpose Returns information about the specified linked-block element.

Return value Return `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description **HDinqblockinfo** operates in a manner similar to **HDinquire** but provides more low-level information than **HLPinquire**. `NULL` can be passed for any non-essential parameters.

HDmalloc

VOIDP HDmalloc(uint32 *quantity*)

quantity IN: Minimum number of bytes to be allocated for the memory block

Purpose Dynamically allocates a block of memory and returns a pointer to it.

Return value A pointer to the allocated memory block if successful, `NULL` otherwise.

Description Operates in a manner similar to **malloc**.

HDmemfill

HDmemfill

VOIDP HDmemfill(VOIDP *dest*, const VOIDP *src*, uint32 *item_size*, uint32 *num_items*)

<i>dest</i>	IN: Pointer to the area in memory to be filled
<i>src</i>	IN: Pointer to the fill pattern
<i>item_size</i>	IN: Size of the fill pattern
<i>num_items</i>	IN: Number of times the pattern will be written into <i>dest</i>

Purpose Fills the memory area pointed to by *dest* with the pattern pointed to by *src* the number of times specified by *num_items*.

Return value A pointer to the *dest* memory buffer.

Description **HDmemfill** can be used to copy a specified fill value into an array of any data type.

The *src* and *dest* parameters are assumed to point to valid memory locations.

HDpackFstring

intn HDpackFstring(char **src*, char **dest*, intn *length*)

<i>src</i>	IN: Source C string
<i>dest</i>	IN: Destination Fortran-77 string
<i>length</i>	IN: Length of the Fortran-77 string

Purpose Converts a C string to a Fortran-77 string by copying.

Return value Returns `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description **HDpackFstring** operates in a manner similar to **HDc2fstr** except that **HDc2fstr** converts in place while this routine converts via a copy operation.

HDputc

HDputc

intn HDputc(uint8 *byte*, int32 *h_id*)

byte IN: Byte to be written

h_id IN: Data element identifier

Purpose Writes one byte to the specified data element.

Return value Returns the byte written to the element if successful and `FAIL` (or `-1`) otherwise

Description **HDputc** calls **Hwrite** to write the byte and reports any error codes returned.

HDrealloc

VOIDP HDrealloc(VOIDP *blk_ptr*, uint32 *quantity*)

<i>blk_ptr</i>	IN:	Pointer to the block of memory to be resized
<i>quantity</i>	IN:	Minimum number of bytes to allocate for the memory block

Purpose	Dynamically reallocates a block of memory and returns a pointer to it.
Return value	A pointer to the reallocated memory block if successful, <code>NULL</code> otherwise.
Description	Operates in a manner similar to realloc .

HDstrdup

HDstrdup

char *HDstrdup(const char *str_ptr*)

str_ptr IN: Pointer to the string to duplicate

Purpose Duplicates a string (i.e. allocates space and copies the string to it).

Return value Pointer to the duplicated string if successful, or `NULL` otherwise.

Description Operates in a manner similar to **strdup**. Designed for use on the following platforms: Macintosh, IBM 6000, VAX, NeXT, MIPSEL and Convex Exemplar.

HDvalidfid

intn HDvalidfid(int32 *value*)

<i>value</i>	IN: 32-bit integer value to be verified
Purpose	Determines whether or not a specified int32 value is a valid HDF file identifier.
Return value	Returns <code>TRUE</code> (or 1) if the specified value is a file identifier and <code>FALSE</code> (or 0) if not.