## HEclear

```
VOID HEclear(void)
```

| | |
|---|---|
| Purpose | Clears all information on reported errors off of the error stack. |
| Return value | None |
| Description | **`HEpush`** creates an error stack. **`HEclear`** is then used to clear this stack after any errors are processed. |
| FORTRAN | None |

# heprnt

## HEprint

```
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 on the error stack.

Return value — None

Description — Prints information on reported errors. If level is zero, 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:

1. ASCII description of the error
2. reporting routine
3. reporting routine as source file name
4. 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 routine uses one less parameter than the C routine because it doesn't allow the user to specify the print stream. Instead it prints automatically to `stdout`.

FORTRAN

```
subroutine heprnt(level)
   integer*4 level
```

## HEpush

```
VOID HEpush(int16 error_code, char *funct_name, char *file_name, intn line)
```

| | | |
|---|---|---|
| *error_code* | IN: | Internal number of the error |
| *funct_name* | IN: | Name of function in which the error occurred |
| *file_name* | IN: | Name of file in which the error occurred |
| *line* | IN: | Line number in the file that error occurred |

Purpose — Pushes a new error onto the error stack.

Return value — None

Description — **HEpush** pushes the file name, function name, line number, and generic description of the error onto the error stack. **HEreport** can then be used to give a more case-specific description of the error.

If the stack is full, the error is ignored.**HEpush** assumes that the character strings *funct_name* and *file_name* are in semi-permanent storage, so only pointers to the strings are saved.

FORTRAN — None

# heprnt

## HEreport

```
VOID HEreport(char *format, ...)
```

*format* IN: `Printf` compatible output string

Purpose Adds a text string to the description of the most recently reported error (only one text string per error).

Return value None

Description By default, an error report made with **`HEpush`** places on the error stack the file name, function name, line number, and a generic description of the error type. `HEreport` can then be used to give a more case-specific description of the error. Only one additional annotation can be attached to each error report.

The arguments to this routine follow the style of `printf`.

Example

```
char *FUNC = "Hclose";
....
if (file_rec->attach > 0) {
    file_rec->refcount++;
    HERROR(DFE_OPENAID);
    HEreport("There are still %d active aids attached",
                    file_rec->attach);
    return FAIL;
}
```

FORTRAN None

## HEstring

```
char *HEstring(int16 error_code)
```

| | | |
|---|---|---|
| *error_code* | IN: | HDF error code |

| | |
|---|---|
| Purpose | Returns the error message associated with error code. |
| Return value | Returns a pointer to a string associated with the error code if successful. |
| Description | Returns a text description for the given error code. These strings are statically declared and should not be `free()ed` by the user. If no string can be found to describe an error code, a generic default message is returned. |
| FORTRAN | None |