

# H5L\_ITERATE1

[Expand all](#) [Collapse all](#)

- [Jump to ...](#)
- [Summary](#)
- [Description](#)
- [Example](#)
- [Switch language ...](#)
- [C](#)
- [C++](#)
- [FORTRAN](#)
- [JAVA](#)

[Summary](#)  
[Description](#)  
[Example](#)  
[JAVA](#)  
[FORTRAN](#)  
[C++](#)  
[C](#)

# H5L\_ITERATE1

Iterates through links in a group

*As of HDF5-1.12 this function has been deprecated in favor of the function [H5L\\_ITERATE2](#) or the macro [H5L\\_ITERATE](#).*

## Procedure:

H5L\_ITERATE1 ( group\_id, idx\_type, order, idx\_p, op, op\_data )

## Signature:

```
herr_t H5Literate1 ( hid_t group_id, H5_index_t idx_type, H5_iter_order_t order,  
    hsize_t *idx_p, H5L_iterate1_t op, void *op_data )  
)
```

```
SUBROUTINE h5literate_f(group,index_type, order, position,  
    op, ctx, return_value, hdferr)  
    INTEGER(HID_T) , INTENT(IN) :: group  
    INTEGER , INTENT(IN) :: index_type  
    INTEGER , INTENT(IN) :: order  
    INTEGER(HSIZE_T), INTENT(INOUT) :: position  
    TYPE(C_FUNPTR) , INTENT(IN) :: op  
    TYPE(C_PTR) , INTENT(IN) :: ctx  
    INTEGER , INTENT(OUT) :: return_value  
    INTEGER , INTENT(OUT) :: hdferr
```

## Parameters:

---

Name	Mode	Kind	Description
group_id	IN	handle	File or group identifier
idx_type	IN	choice	Index for iteration order
order	IN	choice	Traversal order
idx_p	INOUT	integer	Iteration position
op	IN	delegate	User-defined operator
op_data	INOUT	opaque	User-supplied context

### Description:

H5L\_ITERATE1 iterates through the links in a file or group, `group_id`, in the order of the specified index, `idx_type`, using a user-defined callback routine `op`. H5L\_ITERATE1 does not recursively follow links into subgroups of the specified group.

Three parameters are used to manage progress of the iteration: `idx_type`, `order`, and `idx_p`.

`idx_type` specifies the index to be used. If the links have not been indexed by the index type, they will first be sorted by that index then the iteration will begin; if the links have been so indexed, the sorting step will be unnecessary, so the iteration may begin more quickly.

The choices for `idx_type` are:

Name	Meaning
H5_INDEX_NAME	Lexicographic order
H5_INDEX_CRT_ORDER	Creation order

`order` specifies the order in which objects are to be inspected along the index `idx_type`. The choices for `order` are:

Name	Meaning
H5_ITER_INC	Increasing order
H5_ITER_DEC	Decreasing order
H5_ITER_NATIVE	Fastest order

`idx_p` tracks the iteration and allows an iteration to be resumed if it was stopped before all members were processed. It is passed in by the application with a starting point and returned by the library with the point at which the iteration stopped.

The `op` callback function, the related `H5L_info1_t` struct, and the effect of the callback function's return value on the application are described in H5L\_VISIT.

`op_data` is a user-defined pointer to the data required to process links in the course of the iteration. This pointer is passed back to each step of the iteration in the `op` callback function's `op_data` parameter.

`op` is invoked for each link encounter:

#### Signature:

```
herr_t (*H5L_iterate1_t)
(
    hid_t          group,
    const char*    name,
    const H5L_info1_t* info,
    void*         op_data
)
```

#### Info:

```

typedef struct {
    H5L_type_t    type;          /* Type of link */
    hbool_t       corder_valid; /* Indicates whether creation
                                /* order is valid */
    int64_t       corder;        /* Creation order */
    H5T_cset_t    cset;          /* Character set of link name */
    union {
        haddr_t   address;       /* Address hard link points to */
        size_t     val_size;     /* Size of soft link or
                                /* user-defined link value */
    } u;
} H5L_infol_t;

```

**Returns:**

Return value	Meaning
0	Continue iteration
> 0	Stop iteration; successful termination
< 0	Stop iteration; abnormal termination

op\_data is passed to and from each iteration and can be used to supply or aggregate information across iterations.

The behavior of H5L\_ITERATE1 is undefined if the link membership of group\_id changes during the iteration. This does not limit the ability to change link destinations while iterating, but caution is advised.

For recursive behavior, see the following:

- [H5L\\_VISIT1](#)
- [H5L\\_VISIT\\_BY\\_NAME1](#)
- [H5O\\_VISIT](#)
- [H5O\\_VISIT\\_BY\\_NAME](#)

**Returns:**

On success, returns the return value of the first operator that returns a positive value, or zero if all members were processed with no operator returning non-zero.

On failure, returns a negative value if something goes wrong within the library, or the first negative value returned by an operator.

**Example:**

```

1_10 / C / H5G / h5ex_g_iterate.c      master      HDFFV/hdf5-examples
/*****

This example shows how to iterate over group members using
H5Literate.

This file is intended for use with HDF5 Library version 1.8

*****/

#include "hdf5.h"

```

```

#include <stdio.h>

#define FILE          "h5ex_g_iterate.h5"

/*
 * Operator function to be called by H5Literate.
 */
herr_t op_func (hid_t loc_id, const char *name, const H5L_info_t *info,
                void *operator_data);

int
main (void)
{
    hid_t          file;          /* Handle */
    herr_t          status;

    /*
     * Open file.
     */
    file = H5Fopen (FILE, H5F_ACC_RDONLY, H5P_DEFAULT);

    /*
     * Begin iteration.
     */
    printf ("Objects in root group:\n");
    status = H5Literate (file, H5_INDEX_NAME, H5_ITER_NATIVE, NULL, op_func, NULL);

    /*
     * Close and release resources.
     */
    status = H5Fclose (file);

    return 0;
}

/*****

Operator function.  Prints the name and type of the object
being examined.

*****/
herr_t op_func (hid_t loc_id, const char *name, const H5L_info_t *info,
                void *operator_data)
{
    herr_t          status;
    H5O_info_t      infobuf;

    /*
     * Get type of the object and display its name and type.
     * The name of the object is passed to this function by
     * the Library.
     */
    status = H5Oget_info_by_name (loc_id, name, &infobuf, H5P_DEFAULT);
    switch (infobuf.type) {
        case H5O_TYPE_GROUP:
            printf (" Group: %s\n", name);
            break;
        case H5O_TYPE_DATASET:

```

```
        printf (" Dataset: %s\n", name);
        break;
    case H5O_TYPE_NAMED_DATATYPE:
        printf (" Datatype: %s\n", name);
        break;
    default:
        printf ( " Unknown: %s\n", name);
}
```

```
    return 0;
}
```

1\_8 / FORTRAN / H5G / h5ex\_g\_iterate\_F03.f90

master H5FFV/h

df5-examples

```
!*****
!  
! This example shows how to iterate over group members using
! H5Literate.
!  
! This file is intended for use with HDF5 Library version 1.8
! with --enable-fortran2003
!  
!  
!*****
MODULE g_iterate

    USE HDF5
    USE ISO_C_BINDING
    IMPLICIT NONE

CONTAINS

!*****
!  
! Operator function. Prints the name and type of the object
! being examined.
!  
! *****

INTEGER FUNCTION op_func(loc_id, name, info, operator_data) bind(C)

    USE HDF5
    USE ISO_C_BINDING
    IMPLICIT NONE

    INTEGER(HID_T), VALUE :: loc_id
    CHARACTER(LEN=1), DIMENSION(1:10) :: name ! must have LEN=1 for bind(C) strings
    TYPE(C_PTR) :: info
    TYPE(C_PTR) :: operator_data

    INTEGER    :: status, i, len

    TYPE(H5O_info_t), TARGET :: infobuf
    TYPE(C_PTR) :: ptr
    CHARACTER(LEN=10) :: name_string

    !
    ! Get type of the object and display its name and type.
    ! The name of the object is passed to this FUNCTION by
    ! the Library.
    !

    DO i = 1, 10
```

```

        name_string(i:i) = name(i)(1:1)
ENDDO

CALL H5Oget_info_by_name_f(loc_id, name_string, infobuf, status)

! Include the string up to the C NULL CHARACTER
len = 0
DO
    IF(name_string(len+1:len+1).EQ.C_NULL_CHAR.OR.len.GE.10) EXIT
    len = len + 1
ENDDO

IF(infobuf%type.EQ.H5O_TYPE_GROUP_F)THEN
    WRITE(*,*) "Group: ", name_string(1:len)
ELSE IF(infobuf%type.EQ.H5O_TYPE_DATASET_F)THEN
    WRITE(*,*) "Dataset: ", name_string(1:len)
ELSE IF(infobuf%type.EQ.H5O_TYPE_NAMED_DATATYPE_F)THEN
    WRITE(*,*) "Datatype: ", name_string(1:len)
ELSE
    WRITE(*,*) "Unknown: ", name_string(1:len)
ENDIF

op_func = 0 ! return successful

END FUNCTION op_func

END MODULE g_iterate

PROGRAM main

USE HDF5
USE ISO_C_BINDING
USE g_iterate

IMPLICIT NONE

CHARACTER(LEN=17), PARAMETER :: filename = "h5ex_g_iterate.h5"
INTEGER(HID_T) :: file ! Handle
INTEGER :: status
TYPE(C_FUNPTR) :: funptr
TYPE(C_PTR) :: ptr
INTEGER(hsize_t) :: idx
INTEGER :: ret_value
!
! Initialize FORTRAN interface.
!
CALL h5open_f(status)
!
! Open file.
!
CALL H5Fopen_f(filename, H5F_ACC_RDONLY_F, file, status)
!
! Begin iteration.
!
WRITE(*,'(A)') "Objects in root group:"

idx = 0
funptr = C_FUNLOC(op_func) ! call back function

```

```
ptr    = C_NULL_PTR

CALL H5Literate_f(file, H5_INDEX_NAME_F, H5_ITER_NATIVE_F, idx, funptr, ptr,
ret_value, status)

!
! Close and release resources.
!
CALL H5Fclose_f(file, status)
```



```
END PROGRAM main
```

### History:

Release	Change
1.12.0	Function was renamed from H5L_ITERATE and deprecated.
1.8.8	Fortran subroutine added.
1.8.0	C function introduced.

--- Last Modified: March 12, 2020 | 01:55 PM