```
    /* read description length and description from file */
    length = DFANgetfdslen(dfile, FIRST);
    iret = DFANgetfds(dfile,indescr, MAXDESCLEN, FIRST);
    printf("\n\nDescription: \n%s\n" indescr);

    DFclose(dfile);
}
```

Remarks:

- These annotations are associated with the file, not with any particular object within the file.
- We use the general purpose routines `DFopen` and `DFclose`. These routines do not open and close HDF files for you. You must do it explicitly.

  The value `DFACC_READ` is defined in `df.h`. Hence the "`#include df.h`" in the C program. It is assumed that the FORTRAN cannot perform such an include, so `DFACC_READ` is defined with a `PARAMETER` statement.

Getting Annotation Information from a File

DFANlastref

FORTRAN:

```
integer DFANlastref()
```

C:

```
int DFANlastref()
```

Purpose: To return the most recent reference number of a written or read annotation.

Returns: The reference number on success; –1 on error.

* DFANlastref is callable only by C routines. There is no equivalent FORTRAN routine in the HDF library.

```
     $           NOTFIRST    = 0
     $           CR          = char(10))

C****||***** read all file IDs from file ***************

      dfile = DFopen(filename, DFACC_READ, 0)
C     *** first ID ***
      length = DFANgetfidlen(dfile, FIRST)
      ret = DFANgetfid(dfile,inlabel, MAXLABLEN, FIRST)

C     *** rest of IDs ***
      do 200 while ( ret .ge. 0)
          print *,'Length: ',length,'Ret:',ret,'Label:',inlabel
          length = DFANgetfidlen(dfile, NOTFIRST)
          ret = DFANgetfid(dfile,inlabel, MAXLABLEN, NOTFIRST)
  200 continue
      print *, '*** End of file IDs ***'

C     *** read file description length and description ***
      length = DFANgetfdslen(dfile, FIRST)
      print *, 'Description length: ', length
      ret = DFANgetfds (dfile, indescr, MAXDESCLEN, 1)

      print *, 'Description:',CR,indescr
      print *, '*** End of description ***',CR

      ret = DFclose(dfile)
...
```

Figure 5.7 Reading File IDs and a File Description (Continued)

C:

```
/************************************************************
*
* Example: retrieving file IDs and descriptions.
* Opens an HDF file and reads back file IDs and descriptions.
*
************************************************************/

#include "df.h"

#define MAXLABLEN 80
#define MAXDESCLEN 1000
#define FIRST      1
#define NOTFIRST   0
main ()
{
    DF *dfile;
    int ret, length;
    char inlabel[MAXLABLEN+1],indescr[MAXDESCLEN+1];

    /* open file to read file IDs and file description */
    dfile = DFopen("myfile", DFACC_READ, 0);

    /*  read all file IDs from file */
    length = DFANgetfidlen(dfile, FIRST);
    while (length >= 0) {
        ret = DFANgetfid(dfile,inlabel, MAXLABLEN, FIRST);
        printf("\nLabel: %s" inlabel);
        length = DFANgetfidlen(dfile, FIRST);
    }
```

Returns: Length of file ID on success; –1 on failure.

DFANgetfds

FORTRAN:

```
INTEGER FUNCTION DFANgetfds(file,desc, maxlen, isfirst)

INTEGER      file        - pointer to HDF file
CHARACTER*(*) desc       - desc to read from file
INTEGER      maxlen;     - max allowable length for DESC
INTEGER      isfirst;    - 1: first one; 0: next one
```

C:

```
int DFANgetfds(dfile, desc, maxlen, isfirst)

DF   *dfile;          /* pointer to HDF file */
char *desc;           /* desc to read from file */
int maxlen;           /* max allowable length for DESC */
int isfirst;          /* 1: first one; 0: next one */
```

Purpose: To get the file desc from a file.

Returns: Length of file `DESC` on success; –1 on failure.

Example: Reading File Annotations from an HDF file

The example in Figure 5.7 illustrates the use of `DFANgetfidlen`, `DFANgetfid`, `DFANgetfdslen` and `DFANgetfds` to read from an HDF file all file IDs and one file description.

Figure 5.7 Reading File IDs and a File Description

FORTRAN:

```
      program annotate_test

C Program to tread file IDs and file descriptions

C****||**************************************************

      integer dfile, i, ret, first, length
      character*64  filename
      character*10  inlabel
      character*400 indescr

      integer DFopen, DFclose
      integer DFANgetfid, DFANgetfds
      integer DFANgetfidlen,  DFANgetfdslen

      integer DFACC_READ
      integer MAXLABLEN, MAXDESCLEN
      character*1 CR

      parameter (
     $           MAXLABLEN   =10,
     $           MAXDESCLEN  =400,
     $           FIRST       = 1,
```

DFANgetfidlen

```
FORTRAN:
INTEGER FUNCTION DFANgetfidlen(file, isfirst)

INTEGER  file            - pointer to HDF file
INTEGER  isfirst         - 1: first one; 0: next one
```

```
C:
int DFANgetfidlen(dfile, isfirst)

DF   *dfile;          /* pointer to HDF file */
int   isfirst;        /* 1: first one; 0: next one */
```

Purpose: To get the length of a file ID.

Returns: Length of file ID on success; –1 on failure.

DFANgetfid

```
FORTRAN:
INTEGER FUNCTION DFANgetfid(file,id, maxlen, isfirst)

INTEGER      file        - pointer to HDF file
CHARACTER*(*) id         - id to read from file
INTEGER     maxlen;      - max allowable length for ID
INTEGER     isfirst;     - 1: first one; 0: next one
```

```
C:
int DFANgetfid(dfile, id), maxlen, isfirst)

DF   *dfile;          /* pointer to HDF file */
char *id;             /* id to read from file */
int maxlen;           /* max allowable length for ID */
int isfirst;          /* 1: first one; 0: next one */
```

Purpose: To get a file id from a file.

Returns: Length of file ID on success; –1 on failure.

DFANgetfdslen

```
FORTRAN:
INTEGER FUNCTION DFANgetfdslen(file, isfirst)

INTEGER  file            - pointer to HDF file
INTEGER  isfirst         - 1: first one; 0: next one
```

```
C:
int DFANgetfdslen(dfile, isfirst)

DF   *dfile;          /* pointer to HDF file */
int   isfirst;        /* 1: first one; 0: next one */
```

Purpose: To get the length of a file ID.

```
main ()
{
...
    DF *dfile;
    char outlabel[MAXLABLEN+1], outdescr[MAXDESCLEN+1];

    dfile = DFopen("myfile", DFACC_WRITE, 0);

    /* store a file ID in the file*/
    strcpy (outlabel, "File #1");
    DFANaddfid(dfile, outlabel);

    /* get and store description in file */
    /* (assume getdescr stores a description in outdescr) */
    getdescr(outdescr);
    DFANaddfds(dfile, outdescr, strlen(outdescr));

    DFclose(dfile);
...
}
```

Remarks:

- These annotations are associated with the file, not with any particular object within the file.
- We use the general purpose routines `DFopen` and `DFclose`. These routines do not open and close HDF files for you. You must do it explicitly.

  The value `DFACC_WRITE` is defined in `df.h`., hence the "`#include df.h`" in the C program. It is assumed that the FORTRAN cannot perform such an include, so `DFACC_WRITE` is defined with a `PARAMETER` statement.
- Only one label and one description is added in this example, but more could be added if desired.
- In the FORTRAN version the string length for the label should be close to the actual expected string length, because in FORTRAN string lengths generally are assumed to be the declared length of the array that holds the string.

  This is not the case for descriptions, because one of the parameters to `DFANaddfds` is the length of the string.

Reading Annotations for HDF Files

These routines are for reading annotations for HDF files, rather than for HDF objects within HDF files.

NOTE: The file annotation routines require that files be identified by a file pointer, rather than by a filename. This means that you must open the HDF file within your program before calling them. You should also close the HDF file after calling them. The sample code in the example at the end of this section illustrates the use of `DFopen` and `DFclose`.

Example: Adding File Annotations to an HDF File

The example in Figure 5.6 illustrates the use of `DFANaddfid` and `DFANaddfds` to write to an HDF file a label and description for the file itself.

Figure 5.6 Adding a File ID and a File Description.t

FORTRAN:

```
      program annotate_test

C Program to  write a file ID and file descriptions

C****||*************************************************

      integer dfile, ret
      character*64 filename
      character*7  baselabel
      character*10 outlabel
      character*400 outdescr

      integer DFopen, DFclose
      integer DFANaddfid, DFANaddfds

      integer DFACC_WRITE
      integer MAXLABLEN, MAXDESCLEN

      parameter (
     $           DFAN_LABEL  = 0,
     $           DFAN_DESC   = 1,
     $           MAXLABLEN   =10,
     $           MAXDESCLEN  =400)


C****||************ store file ID in file **************

      print *, 'Enter HDF file name:'
      read *, filename

      dfile = DFopen(filename, DFACC_WRITE, 0)
      outlabel = 'Label #1'
      ret = DFANaddfid (dfile, outlabel)

C****||***** get and store file description in file ****
C****||** (assume getdescr stores a description in outdescr) **

      call getdescr(outdescr)
      ret = DFANaddfds (dfile, outdescr,len(outdescr))

      ret = DFclose(dfile)

...
```

Figure 5.6 Adding a File ID and a File Description.t (Continued)

C:

```
#include "df.h"
#define MAXLABLEN 80
#define MAXDESCLEN 1000
```

Writing Annotations for HDF Files

These routines are for writing annotations for HDF files, rather than for HDF objects within HDF files.

NOTE: The file annotation routines require that files be identified by a file pointer, rather than by a filename. This means that you must open the HDF file within your program before calling them. You should also close the HDF file after calling them. The sample code in the example at the end of this section illustrates how to open and close files using `DFopen` and `DFclose`.

DFANaddfid

FORTRAN:

```
INTEGER FUNCTION DFANaddfid(file,id)

INTEGER      file        - pointer to HDF file
CHARACTER*(*) id         - id to write to file
```

C:

```
int DFANaddfid(dfile, id)

DF   *dfile;          /* pointer to HDF file */
char *id;             /* id to write to file */
```

Purpose: To add a file id to a file.

Returns: 0 on success; –1 on failure.

DFANaddfds

FORTRAN:

```
INTEGER FUNCTION DFANaddfds(file, desc, desclen)

INTEGER       file       - pointer to HDF file
CHARACTER*(*) desc       - description to write to file
INTEGER       desclen    - length of description
```

C:

```
int DFANaddfds(dfile, desc, desclen)

DF   *dfile;          /* pointer to HDF file */
char *desc;           /* description to write to file */
int32 desclen;        /* length of description */
```

Purpose: To add a file description to a file.

Returns: 0 on success; –1 on failure.

```
      integer i, nlabels, startpos, listlen
      integer reflist(20)
      integer DFTAG_SDG, , LISTSIZE, MAXLEN
      character*15  labellist(20)

      parameter (DFTAG_SDG   = 700,
     *           LISTSIZE    =  20,
     *           MAXLEN      =  15 )

      startpos = 1

      print *, 'Labels of scientific datasets in myfile'
      nlabels = dallist('myfile',DFTAG_SDG, reflist,
     *                  labellist, LISTSIZE, MAXLEN, startpos)

      do 100 i=1,nlabels
          print *,'  Ref number: ',reflist(i),'
                                Label: ',labellist(i)
  100 continue

      stop
      end
```

C:

```
/*
* Program to test getlablist routine
*/

#include <stdio.h>
#include "df.h"
char *malloc();

#define LISTSIZE 20
#define MAXLEN   15

main()
{
    int i, nlabels, startpos=1, listlen=10;
    uint16 reflist[LISTSIZE];
    char labellist[MAXLEN*LISTSIZE+1];

    printf("Labels of scientific datasets in myfile");
    nlabels = DFANlablist("myfile",DFTAG_SDG, reflist,
                        labellist, listlen, MAXLEN, startpos);
    for (i=0; i<nlabels; i++)
        printf("\n\t%d\tRef number: %d\tLabel: %s",
                            i, reflist[i],labellist+(i*15));
    printf("\n\n");
}
```

Remarks:

- In the call to `DFANlablist (dallist)` you must pre-allocate the arrays, `reflist` and `labellist`, with enough space to hold the reference numbers and labels, respectively.

- The `LISTSIZE` in the parameter list tells the routine to read, at most, 20 labels and reference numbers. The `MAXLEN` indicates that each label is to be stored in a 15-byte sequence in the array `labellist`. Since `startpos=1`, the routine will start with the first entry.

```
CHARACTER*(*) filename    -  name of HDF file labels stored in
CHARACTER*(*) labellist   -  array of strings to place labels in
INTEGER tag               -  tag to find labels for
INTEGER reflist(*)        -  array to place refs in
INTEGER listsize          -  size of ref and label lists
INTEGER maxlen            -  maximum length allowed for label
INTEGER startpos          -  entries will be returned beginning
            from the startpos entry up to the
listsize entry.
```

C:

```
int DFANlablist(filename, tag, reflist, labellist, listsize, maxlen,
startpos)

char *filename;       /* name of HDF file labels stored in */
uint16 tag;           /* tag to find labels for */
unit16 reflist[];     /* array to place refs in */
char *labellist;      /* array of strings to place labels in */
int listsize;         /* size of ref and label lists */
int maxlen;           /* maximum length allowed for label */
int startpos;         /* entries will be returned beginning from
   the startpos entry up to the listsize             entry */
```

Purpose: To return a list of all reference numbers and labels for a given tag.

Returns: The number of entries on success; –1 on error.

Input parameters are `filename`, `tag`, `listsize`, `maxlen` and `startpos`. *`Listsize`* gives the number of available entries in the ref and label lists, *`maxlen`* is the maximum length allowed for a label, and *`startpos`* tells which label to start reading for the given tag. Beginning from the `startpos` entry and extending to the `listsize` parameter, entries will be returned. (If `startpos` is 1, for instance, all labels will be read; if `startpos`=4, all but the first 3 labels will be read.)

Taken together, the `reflist` and `labellist` returned by `DFANlablist` constitute a directory of all labels for a given tag in a file. The list, *`labellist`*, can be displayed to show all of the labels for a given tag. Or, it can be searched to find the ref of a data object with a certain label. Once the ref for a given label is found, the corresponding data object can be accessed by invoking the routine `DFgetelement`. This routine provides you with a mechanism for direct access to data objects in HDF files.

Example: Getting a List of Labels for Images in a File

The example in Figure 5.5 illustrates the use of `DFANlablist` to get a list of all labels used for SDSs in an HDF file.

Figure 5.5 Getting a List of Labels from a File

FORTRAN:

```
      program getlablist

C Program to test getlablist routine
C

      integer dallist
```

Figure 5.4 Getting Annotations from a SDS (Continued)

C:

```
/*
* Program to test routines for reading a label and description
*/
#include "df.h"
extern uint16 DFfindnextref();

main()
{
    int    desclen;
    uint16 ref;
    char   label[20], *s;
    DF     *dfile;       /* HDF file pointer */

    /*** find ref of first SDS in file ***/
    dfile = DFopen("myfile", DFACC_READ, -1);
    ref = DFfindnextref(dfile, (unsigned short) DFTAG_SDG, 0);
    if (ref < 0) {
        printf("Unable to find scientific dataset.\n");
        exit(1);
    }
    DFclose(dfile);

    /*** get label, then description ***/
    DFANgetlabel("myfile", (uint16) DFTAG_SDG, ref, label, 11);
    printf("Label: %s\n", label);

    desclen = DFANgetdesclen("myfile",(uint16) DFTAG_SDG, ref);
    s = malloc( desclen+1);
    DFANgetdesc("myfile", (uint16) DFTAG_SDG, ref, s, desclen);
    printf("Description: %s\n", s);
}
```

Remarks:

- Lower level routines `DFopen`, `DFclose`, and `DFfindnextref` are used here to find the ref number of the first occurrence of the SDG (scientific data group) tag.
- In the above example, `DFANgetlabel` assumes that the label is not more than 10 bytes long. If the program needs to know the length of a label, it can call `DFANgetlablen` to find this out.
- Since the description could be very long, the routine `DFANgetdesclen` is called to find the space requirements for the description. This space is allocated before calling `DFANgetdesc` to get the description.

Listing All Labels for a Given Tag

DFANlablist

FORTRAN:

```
INTEGER FUNCTION DFANlablist(filename, tag, reflist, labellist,
listsize, maxlen, startpos)
```

Figure 5.4 Getting Annotations from a SDS

FORTRAN:

```
      program getanntest

C Program to test routines for reading a label and description

C
C*************************************************************

      integer DFopen, DFclose
      integer dfindnr, daglab, dagdesc, dagdlen
      integer desclen, ref, ret, dfile
      integer DFACC_READ, DFTAG_SDG
      character*20  label
      character*400 desc

      parameter (DFACC_READ  =   1,
     $           DFTAG_SDG   = 700 )


C************** find ref of first SDS in file ***************

      dfile = DFopen('myfile', DFACC_READ, -1)
      ref = dfindnr(dfile, DFTAG_SDG, 0)
      if (ref .lt. 0)
     *  call fatalerror('Unable to find scientific dataset.')
      ret = DFclose(dfile)

C************** get label, then description ***************
      ret = daglab('myfile', DFTAG_SDG, ref, label, 11)
      print *,'Label: ', label

      desclen = dagdlen('myfile', DFTAG_SDG, ref)
      if (desclen .gt. 400)
     $    call fatalerror('Descript too long. More than 400.')
      ret = dagdesc('myfile', DFTAG_SDG, ref, desc, desclen)
      print *,'Description: '
      print *, desc

      stop
      end

C************************************************************
* fatal error: subroutine to report fatal error and abort
C************************************************************

      subroutine fatalerror(s)
      character*(*) s

      print *, s
      print *, 'DFerror:', DFerrno()
      print *, 'Program aborted.'
      print *, ' '
      stop
      end
```

DFANgetdesclen

FORTRAN:

```
INTEGER FUNCTION DFANgetdesclen(filename, tag, ref)

CHARACTER*(*) filename    -  name of HDF file descr is stored in
INTEGER tag, ref          -  tag/ref of item whose descr you
            want
```

C:

```
int32 DFANgetdesclen(filename, tag, ref)

char *filename;       /* name of HDF file descr is stored in */
uint16 tag, ref;      /*tag/ref of item whose descr you want */
```

Purpose: To get the length of a description of the data object with the given tag and reference number. This routine allows you to insure that there is enough space allocated for a description before actually loading it.

Returns: The length of description on success; –1 on failure.

DFANgetdesc

FORTRAN:

```
INTEGER FUNCTION DFANgetdesc(filename, tag, ref, desc, maxlen)

CHARACTER*(*) filename    -  name of HDF file descr is stored in
CHARACTER*(*) desc        -  space to return description in
INTEGER tag, ref          -  tag/ref of item whose descr you
            want
INTEGER maxlen            -  size of space to return descr in
```

C:

```
int DFANgetdesc(filename, tag, ref, desc, maxlen)

char *filename;       /* name of HDF file descr is stored in */
uint16 tag, ref;      /*tag/ref of item whose descr you want */
char *desc;           /*space to return description in */
int32 maxlen;         /*size of space to return descr in */
```

Purpose: To read in the description of the data object with the given tag and reference number.

Returns: 0 on success; –1 on failure.

The parameter *maxlen* gives the amount of space that is available for storing the *description*. The length of maxlen must be at least one greater than the anticipated length of the description, because a NULL byte is appended to the annotation.

Example: Reading a Label and Description
This example program (Figure 5.4) illustrates the use of DFANgetlabel, DFANgetdesclen, and DFANgetdesc to read from an HDF file a label and description for a SDS.

write either one or both, depending on your needs.

Reading Annotations for HDF Objects

DFANgetlablen

FORTRAN:

```
INTEGER FUNCTION DFANgetlablen(filename, tag, ref)

CHARACTER*(*) filename    -  name of HDF file label is stored in
INTEGER tag, ref,         -  tag/ref of item whose label you
             want
```

C:

```
int32 DFANgetlablen(filename, tag, ref)

char *filename;       /* name of HDF file label is stored in */
uint16 tag, ref;      /* tag/ref of item whose label you want */
```

Purpose: To get the length of a label of the data object with the given tag and reference number. This routine allows you to insure that there is enough space allocated for a label before actually loading it.
Returns: The length of label on success; –1 on failure.

DFANgetlabel

FORTRAN:

```
INTEGER FUNCTION DFANgetlabel(filename, tag, ref, label, maxlen)

CHARACTER*(*) filename,   -  name of HDF file label is stored in
CHARACTER*(*) label       -  space to return label in
INTEGER tag, ref          -  tag/ref of item whose label you
             want
INTEGER maxlen            -  size of space to return label in
```

C:

```
int DFANgetlabel(filename, tag, ref, label, maxlen)

char *filename;       /* name of HDF file label is stored in */
uint16 tag, ref;      /* tag/ref of item whose label you want */
char *label;          /* space to return label in */
int32 maxlen;         /* size of space to return label in */
```

Purpose: To read in the label of the data object with the given tag and reference number.

Returns: 0 on success; –1 on failure.

The parameter *maxlen* gives the amount of space that is available for storing the *label*. The length of maxlen must be at least one greater than the anticipated length of the label, because a NULL byte is appended to the annotation.

Returns: 0 on success; –1 on failure.

The parameter `desclen` gives the length of the description that is to be written out.

Example: Adding Annotations to a Scientific Dataset

The example in Figure 5.3 illustrates the use of `DFANputlabel` and `DFANputdesc` to write to an HDF file a label and description for a SDS. The HDF object that contains a SDS is call a scientific data group . Essentially, a *Scientific Data Group* (SDG) is a group of tag/refs that make up a SDS. The tag for a scientific data group is `DFTAG_SDG`.

Figure 5.3 Adding Annotations to a SDS

FORTRAN:

```
integer dsadata, daplab, dapdesc, dslref
integer ret, lref
parameter (DFTAG_SDG = 700)
real*4 dataset(2,5)
...
ret = dsadata('myfile',2,shape,dataset)

lref = dslref()
ret = daplab('myfile', DFTAG_SDG, lref,'testlab')
Ret = dapdesc('myfile', DFTAG_SDG, lref,'This is a test',14)
```

C:

```
#include "df.h"
...
int i, rank, dimsizes[2];
char s[50];
float *data;
...
DFSDadddata("myfile",rank,dimsizes,data);
...
sprintf(s,"Data from black hole experiment\n8/18/87");
i = DFSDlastref();
DFANputlabel("myfile", DFTAG_SDG, i, "black hole");
DFANputdesc("myfile", DFTAG_SDG, i, s, strlen(s));
```

Remarks:

- `DFTAG_SDG` is the tag that goes with a scientific data group.
- The call to `DFSDlastref (dslref)` returns the reference number of the SDS last written to the file. This call is needed to complete the tag/ref combination that uniquely identifies the desired scientific data group that is being annotated.
- The file `df.h` that is included with the C source contains the number that corresponds to the tag for a SDS. In the FORTRAN program, this number is defined using a parameter statement. Tags and their numbers are listed in Appendix A, "NCSA HDF Tags."
- It is not necessary to write both a label and a description. You can

DFANgetfid	**dagfid**	gets file ID.
DFANgetfdslen	**dagfdsl**	gets file description length.
DFANgetfds	**dagfds**	gets file description.
DFANlastref*		returns ref of last annotation read or written.

Writing Annotations for HDF Objects

DFANputlabel

FORTRAN:

```
INTEGER FUNCTION DFANputlabel(filename, tag, ref, label)

CHARACTER*(*) filename    - name of HDF file to put label in
CHARACTER*(*) label       - label to write to the file
INTEGER tag, ref          - tag/ref of item whose label we want
                            to store
```

C:

```
int DFANputlabel(filename, tag, ref, label)

char *filename;       /* name of HDF file to put label in */
uint16 tag, ref;      /* tag/ref of item whose label you want to
                         store*/
char *label;          /* label to write to the file */
```

Purpose: To write out a label for the data object with the given tag/ref.

Returns: 0 on success; –1 on failure.

DFANputdesc

FORTRAN:

```
INTEGER FUNCTION DFANputdesc(filename, tag, ref, desc, desclen)

CHARACTER*(*) filename    - name of HDF file to put descr in
CHARACTER*(*) desc        - description to write to the file
INTEGER tag, ref          - tag/ref of item whose description
                            you want to store
INTEGER desclen           - length of description
```

C:

```
int DFANputdesc(filename, tag, ref, desc, desclen)

char *filename;       /* name of HDF file descr stored in */
uint16 tag, ref;      /* tag/ref of item whose descr you want to
                          store */
char *desc;           /* description to write to file */
int32 desclen;        /* length of description */
```

Purpose: To write out a description for the data object with the given tag/ref.

in either reading or writing the corresponding data object.

Reference numbers for objects other than these can be obtained with the routine *DFfindnextref*, a general purpose HDF routine that requires the use of the HDF routines DFopen and DFclose. Usage of DFfindnextref is illustrated later in an example. See this chapter's section, "Example: Reading a Label and Description."

The Annotation Interface

The HDF library provides two types of routines for storing and retrieving annotations: (1) routines for file IDs and file descriptions, and (2) routines for HDF data objects. These routines are callable from C and FORTRAN programs that have access to the HDF library, version 3.0 and later.

In the following explanations, the term *label* refers to a string that identifies a data element such as an image or floating-point dataset. A label is a C string: it can contain any sequence of ASCII characters except NULL, which terminates the string. A *description*, on the other hand, can contain any sequence of ASCII characters, including NULL. Hence, a description can contain several C strings. In those routines that read or write descriptions, it is always necessary to specify explicitly the lengths of the descriptions.

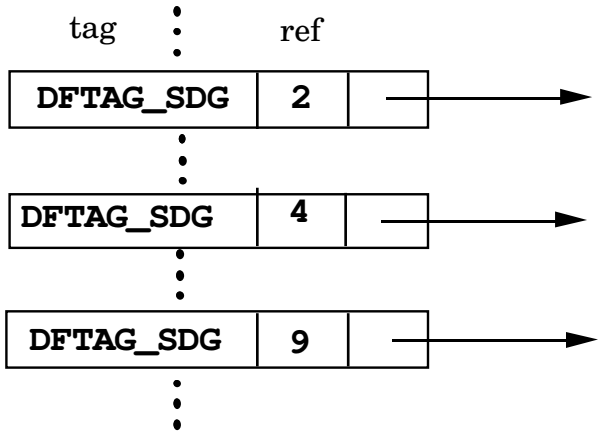
All of the callable annotation routines begin with the letters DFAN. Since some FORTRAN compilers only accept identifiers with eight or fewer characters, we've devised an alternate set of short names that you can use when programming with one of these compilers. Table 5.1 contains the normal names of the annotation routines, as well as the shorter names. Both sets of names are supported on all HDF-supported machines.(Refer to Appendix E, "Routine Lists," for a complete listing of HDF routines.)

Table 5.1 Long and Short Names for Annotation Routines

Long Name	Short Name	Purpose
DFANputlabel	**daplab**	puts label of tag/ref.
DFANputdesc	**dapdesc**	puts description of tag/ref.
DFANgetlablen	**dagllen**	gets length of label of tag/ref.
DFANgetlabel	**daglab**	gets label of tag/ref.
DFANgetdesclen tag/ref.	**dagdlen**	gets length of description of
DFANgetdesc	**dagdesc**	gets description of tag/ref.
DFANlablist	**dallist**	gets list of labels for a particular tag.
DFANaddfid	**daafid**	adds file ID .
DFANaddfds	**daafds**	adds file description.
DFANgetfidlen	**dagfidl**	gets file ID length.

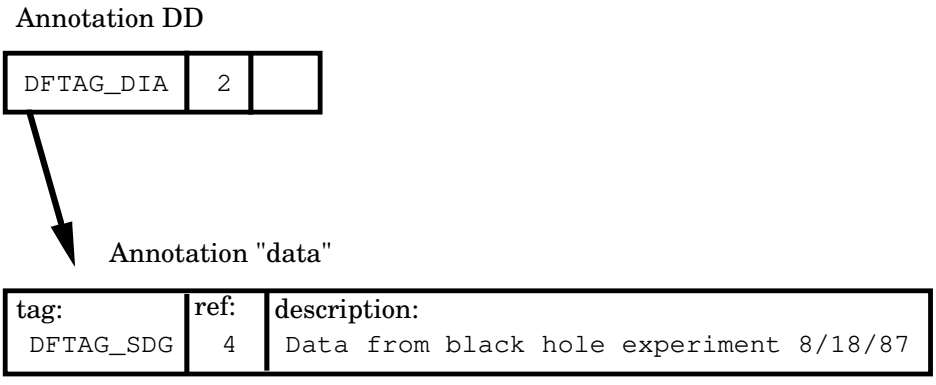
For example, suppose you have an HDF file that contains three scientific datasets (SDS). Each SDS has its own DD consisting of the SDS tag `DFTAG_STG`, and a unique reference number as illustrated in Figure 5.1.

Figure 5.1 Three SDS Tags with Their Ref Numbers



Suppose you wish to annotate the second SDS by storing the following annotation with it in the file: "`Data from black hole experiment 8/18/87`." This text would be stored in an HDF file as an annotation, and it would have stored with it the tag `DFTAG_SDG` and reference number 4. Figure 5.2 illustrates how the annotation would look in the file.

Figure 5.2 Displayed Example of SDS, Ref #, and Annotation



Tags and Reference Numbers

Note that in order to use annotation routines, you need to know the tags and reference numbers of the objects you wish to annotate. Tags are listed in Appendix A, "NCSA HDF Tags."

Special routines are available for obtaining the reference numbers of certain tags, including tags for SDSs, Raster Image Sets, palettes, and annotations. These are: `DFSDlastref`, `DFR8lastref`, `DFPlastref`, and `DFANlastref`. They return the most recent reference number used

Chapter Overview

This chapter describes the routines that are available for storing and retrieving data and file annotations.

Annotation Tags

It is often useful to associate in text form information about an HDF file and its data contents, and to keep that information in the same file that contains the data. HDF provides this capability in the form of annotations. An HDF *annotation* is a sequence of ASCII characters that is associated with one of three types of objects: (1) the file itself, (2) the individual HDF data objects in the file, or (3) the tags that identify the data elements. The current annotation interface supports only the first two types of annotation.

HDF annotations can accommodate a wide variety of types of information, including titles, comments, variable names, parameters, formulas, and source code. Any textual information that a user might normally put into a notebook concerning the collection, meaning, or use of a file or data can be put into a file's annotations.

Annotations are optionally supplied by a creator or user of an HDF file or data object. Annotations come in two forms: *labels*, which normally consist of short strings of characters, and *descriptions*, which can be long and complex bodies of text.

File Annotations

Any HDF file can have labels (called *file IDs*) and descriptions stored in them. There are routines in the annotations interface specifically designed for reading and writing file IDs and file descriptions.

HDF Object Annotations

The annotation of HDF data objects is complicated by the fact that you have to uniquely identify the objects being annotated. In order to understand how annotations work for data objects, you need to know a little bit about how HDF data objects are structured and identified within an HDF file.

HDF *data objects* are the basic building blocks of HDF files. An HDF data object has two parts: a 12-byte *Data Descriptor* (DD) and a *data element*. A DD has four fields: a tag, a reference number, a 32-bit data offset, and a 32-bit data length. (The latter two are unimportant here.) Taken together, the tag and reference number for a data object uniquely identify that object. Hence, the data object that a particular annotation refers to can be identified by storing the object's tag and reference number *together with* the annotation.
Note that an HDF annotation is itself a data object, so it has its own DD. This DD has a tag and a ref number, and it points to the "data" that constitutes the annotation. The "data" that goes with an annotation consists of three things: (1) the tag of the object that it is an annotation for, (2) the ref of the object that it is an annotation for, and (3) the annotation itself.

Chapter 5 Annotating Data Objects and Files