```
yoyodyne_51% mail archive-server@ncsa.uiuc.edu
Subject:  help

.
EOT
Null message body; hope that's ok
yoyodyne_52% mail archive-server@ncsa.uiuc.edu
Subject:  index
.
EOT
Null message body; hope that's ok
```

The information you receive from both the help and index commands will give you further instructions on obtaining NCSA software. This controlled-access server will e-mail the distribution to you one segment at a time.

U.S.Mail

Like other NCSA software, NCSA HDF is also available for purchase—either individually or as part of the anonymous FTP reel or cartridge tapes—through the NCSA Technical Resources Catalog. Orders can only be processed if accompanied by a check in U. S. dollars made out to the University of Illinois. To obtain a catalog, contact:

NCSA Documentation Orders
152 Computing Applications Building
605 East Springfield Avenue
Champaign, IL 61820
(217) 244-0072

Your login session should resemble the sample presented below, where the remote user's local login name is *smith* and user entries are indicated in boldface type.

```
harriet_51%  ftp ftp.ncsa.uiuc.edu
Connected to zaphod.
220 zaphod FTP server (Version 4.173 Tue Jan 31 08:29:00 CST 1989)
ready.
Name (ftp.ncsa.uiuc.edu: smith):  anonymous
331 Guest login ok, send ident as password.
Password:  smith
230 Guest login ok, access restrictions apply.
ftp>  cd HDF
250 CWD command successful
ftp>  get README.FIRST
200 PORT command successful.
150 Opening ASCII mode data connection for README.FIRST (10283
bytes).
226 Transfer complete.
local:  README.FIRST  remote:  README.FIRST
11066 bytes received in .34 seconds (32 Kbytes/s)
ftp>  quit
221 Goodbye.
harriet_52%
```

NCSA HDF documentation, program, and source code are now in the public domain. You may copy, modify, and distribute these files as you see fit.

Archive Server

To obtain NCSA software via an archive server:

1. E-mail a request to:

   archive-server@ncsa.uiuc.edu

2. Include in the subject or message line, the word "**help**."

3. Press RETURN.

4. Send another e-mail request to:

   archive-server@ncsa.uiuc.edu

5. Include in the subject or message line, the word "**index**."

6. Press RETURN.

For example, if you use the UNIX mailing system, your login session should resemble the following sample, where user entries are indicated in boldface type.

HDF file generated by your FORTRAN or C program), then be sure to turn Macbinary mode **off** before performing the transfer.

- If the HDF file corresponds to a Macintosh application (e.g., NCSA Layout, NCSA DataScope, etc.), and you want to transfer it so that it can be accessed from a Macintosh application on another Mac, then be sure to turn Macbinary mode **on** before performing the transfer.

How to Get HDF

You may obtain NCSA software via FTP, an archive server, or U.S. mail. Instructions for doing so are provided below.

FTP

If you are connected to Internet (NSFNET, ARPANET, MILNET, etc.) you may download NCSA HDF software, documentation, and source code, at no charge from an anonymous file transfer protocol (FTP) server at NCSA. The procedure you should follow to do so is presented below. If you have any questions regarding this procedure or whether you are connected to Internet, consult your local system administration or network expert.

1. Log on to a host at your site that is connected to the Internet and is running software supporting the FTP command.

2. Invoke FTP on most systems by entering the Internet address of the server:

   ```
   ftp  ftp.ncsa.uiuc.edu
   ```

   or

   ```
   ftp  128.174.20.50
   ```

3. Log in by entering **`anonymous`** for the name.

4. Enter your local login name for the password.

5. Enter **`cd HDF`** to move to the HDF directory.

6. Enter **`get README.FIRST`** to transfer the instructions (ASCII) to your local host.

7. Enter **`quit`** to exit FTP and return to your local host.

8. Review the README.FIRST file for complete instructions concerning the organization of the FTP directories and the procedure you should follow to download the README files that contain further information on how to get and compile the most recently released version of HDF for your machine and operating system and to determine which files to transfer to your home machine.

FORTRAN 77 and K&R's C

As much as possible, we have tried to stick closely to those implementations of the two languages that are in most common use today, namely FORTRAN 77 and the version of C that is described in Kernighan & Ritchie's *The C Programming Language*, First Edition. If your FORTRAN or C compiler understands FORTRAN 77 or K&R C, it should be able to link easily to the interfaces.

Although we try to adhere to these standards, we must note that a primary objective of the HDF project is to support HDF on a variety of different machines, and this, in some cases, means accommodating some deviations. We are also aware of the fact that many potential users of HDF have compilers that our code does not accommodate. Let us know if your particular dialect does not work with HDF. We may or we may not be able to help.

HDF Without FORTRAN

If you do not use FORTRAN with HDF, you may want to compile the HDF library without any FORTRAN routines in it. Two instances in which you may choose to do so are the following:

1) If you want to reduce the size of the HDF library

2) If you don't have a FORTRAN compiler to use in compiling the library

Details on how to compile HDF without FORTRAN are contained in the INSTALL file, a copy of which is included in Appendix F, "NCSA HDF README Files on Anonymous FTP."

Installing HDF

Details on how to install HDF are beyond the scope of this manual, but you can get quite a bit of information about the process from the readme files that come with the source code. These files are reprinted in Appendix F. See the section below "How to Get HDF" for information on how to get the actual HDF software.

Transferring HDF Files

HDF files are binary files, so any transfer protocol that transfers binary files without changing them can be used to transfer HDF files.

Many HDF users use FTP to transfer HDF files. If you use FTP, switch to binary mode when transferring HDF files.

If you use NCSA Telnet and you wish to transfer an HDF file to or from a Macintosh, you must pay special attention to whether or not to enable the "Macbinary" option. There are two case to consider:

- If the HDF file is not from a Macintosh application (e.g., it is a normal

- `REAL x(*)`
  means that x refers to an array of reals of indefinite size and of *indefinite rank*. It is the responsibility of the calling program to allocate an actual array with the correct number of dimensions and dimension sizes.

Case Sensitivity

Another difference between FORTRAN and C is that FORTRAN identifiers, in general, are not case sensitive, whereas C identifiers are. Hence, although the names of functions and variables listed in this manual use both lower-case and upper-case letters, FORTRAN programs that call them can use either upper or lower case without loss of meaning.

Name Length

Since some FORTRAN compilers can only interpret identifier names with seven or fewer characters, the names of some of the FORTRAN routines are sometimes shorter than the names of the corresponding C routines. (Example: `DFget` (FORTRAN) vs. `DFgetelement` (C)). A listing of short names for all functions can be found in Appendix C, "Eight-Character FORTRAN Names."

Header Files

If you use the high level interfaces, you probably do **not** need to include an HDF header file with your program. However, you may need to include a header file if your program uses special HDF declarations or definitions. This would likely be the case, for instance, if you use the general purpose routines described in Chapter 6, "General Purpose HDF Routines."

There are two header files, one for FORTRAN and one for C:

- **`dfF.h`** contains the declarations and definitions that are used by FORTRAN routines.

- **`df.h`** contains the declarations and definitions that are used by the C routines.

There is also a file called `constants.f` that contains FORTRAN *parameter* statements that declare the HDF constants that you are most likely to use in a FORTRAN program that invokes HDF routines.

For example, if your program uses mnemonics for tags, the corresponding numerical values for the tags can be found in `constants.f` (FORTRAN), `dfF.h` (FORTRAN) or `df.h` (C). The contents of `dfF.h` , `df.h` and `constants.f` are listed in Appendix B, "Header Files."

Although the use of header files is always permitted in C programs, it is not generally permitted in FORTRAN. It is, however, sometimes available as an option in FORTRAN. On UNIX systems, for example, the macro processors `m4` and `cpp` let your compiler include and preprocess header files. If this or a similar capability is not available, you may have to copy whatever declarations, definitions, or values you need from `dfF.h` into your program code.

```
ret = DFSDsetdims(2, shape);
ret = DFSDsetdatastrs("pressure 1",
              "Pascals","E15.9","cartesian");
ret = DFSDsetdimstrs(1,"x","cm","F10.2");
ret = DFSDsetdimstrs(2,"y","cm","F10.2");
ret = DFSDadddata("Ex.hdf", 2, shape, pressure);
```

NOTE: The "set" calls (`DFSDsetdims`(), etc.) indicate the ancillary information that is to be stored with the SDS. `DFSDsetdims` is required; the others are optional. `DFSDadddata` writes the scientific dataset data to `Ex.hdf`. If `Ex.hdf` exists, the SDS is appended to the file. If Ex.hdf does not exist, a new file is created, and the SDS is written as the first in the file. The variable `ret` is assigned the value 0 if `DFSDsetdims`, etc., succeeds; the default value is, -1.

FORTRAN and C

The necessity to support both FORTRAN and C interfaces for HDF inevitably leads to some difficulties in the design of the interfaces. What is natural to a C interface can be quite unnatural to a FORTRAN interface and vice versa. In order to make the FORTRAN and C versions of each routine as identical as possible, some compromises have often had to be made in the simplification of one or the other routine.

FORTRAN Stubs

Another element that affects the appearance of the routines is the fact that almost all of the actual code underlying the HDF interfaces is written in C. Every call to a FORTRAN routine ultimately makes access to a C program that actually carries out the prescribed function. So, the FORTRAN routines might better be referred to as FORTRAN *stubs* rather than FORTRAN functions. When called, these stubs typically translate all parameter values immediately to a data type that is accessible to C, then call a corresponding C function to do the actual work.

Data Type Anomalies

Differences between the two languages also leads to some difficulties in describing, using FORTRAN conventions, some of the data types in the argument lists. For instance, some of the scientific dataset routines place no restrictions on the rank (number of dimensions) that a data array can have. This is perfectly legal in C, but unnatural in FORTRAN. Fortunately, since both C and FORTRAN pass arrays by reference, no problem arises in the actual interface between the FORTRAN calls and the corresponding stubs. The only real problem is in the notation used in this manual to describe the routines *as if* they were actual FORTRAN routines.

As a result, in the declarations contained in the headers of FORTRAN functions, we use the following conventions:

- `CHARACTER*1 x(*)`
  means that x refers to an array that contains an indefinite number of characters. It is the responsibility of the calling program to allocate enough space to hold whatever data is stored in the array.

The routine `DFR8getimage` is available for retrieving images from HDF files. Routines are also available for storing color lookup tables (palettes) with raster images.

Linking to the HDF library

If your FORTRAN or C program makes a call to HDF, it must be linked to the HDF library. You can indicate this in the compile statement, or if a separate linkage step is used, it may be done at that time. On UNICOS at NCSA, this linkage can be performed at compile time with the following statement.

FORTRAN:

```
cf77 -o myprog myprog.f -ldf
```

C:

```
cc -o myprog myprog.c -ldf
```

Writing an HDF Scientific Dataset

An HDF *scientific dataset* (SDS) is a collection in an HDF file of information about scientific data stored as a multi-dimensional regular grid. Each SDS must include the actual data array, its rank (number of dimensions), and its dimensions. Optionally, an SDS can also contain scales to be used along the different axes when interpreting the data, maximum and minimum values of the data, and the coordinate system used to interpret the data. Labels, units, and format specifications for displaying and interpreting the data may also be included.

Below is code presented first in FORTRAN, then in C, that stores a 200 x 200 floating-point array called “pressure” in an SDS in the HDF file, `Ex.hdf`. It also stores labels, units, and formats as part of the same SDS.

FORTRAN:

```
INTEGER dssdims, dssdast, dssdist, dsadata
real    pressure(200,200)
INTEGER shape(2), ret

shape(1) = 200
shape(2) = 200

ret = dssdims(2, shape)
ret = dssdast('pressure 1','Pascals','E15.9','cartesian')
ret = dssdist(1,'x','cm','F10.2')
ret = dssdist(2,'y','cm','F10.2')
ret = dsadata('Ex.hdf', 2, shape, pressure)
```

C:

```
int DFSDsetdims(),DFSDsetdatastrs(),DFSDsetdimstrs(),
       DFSDadddata();
float  pressure[200][200];
int shape[2], ret;

shape[1] = 200;
shape[2] = 200;
```

call these routines, you just link the library to your program at compile time.

Detailed information on how to install and use HDF on specific systems can be found in the documentation that comes with the system-specific versions of the software.

Examples

Writing an HDF 8-Bit Raster Image Set

A typical use of HDF involves preparation of scientific data for visualization as an 8-bit raster image. Using 8-bit raster imaging, the values on a grid of numbers can be represented by color values in a palette of 256 colors.

The following code segments, the first in FORTRAN, the second in C, convert a 200 x 100 floating-point array to an 8-bit raster image, then store the image in an HDF file.

FORTRAN:

```
  CHARACTER*1 image(200,100)
  INTEGER istat, d8aimg

C Other Fortran code goes here

C Convert values in array ivals to character (8-bit) data
  do 10 ix=1,200
     do 10 iy=1,100
        image(ix,iy) = char(ivals(ix,iy))
10 continue

C Write image to an HDF file
  istat = d8aimg('myfile.hdf', image, 200, 100, 0)
  if (istat .ne. 0) then
     write(*,*) 'Error writing HDF file'
  endif
```

C:

```
char image[200][100];
int ix, iy, istat, DFR8addimage();

/* Other C code goes here */

for (ix=0; ix<200; ix++)
  for (iy=0; iy<100; iy++)
    image[ix][iy] = (char) (ivals[ix][iy]);

istat = DFR8addimage("myfile.hdf", image, 200, 100, 0);
if (istat != 0)
  printf("Error writing HDF file\n");
```

NOTE: `DFR8addimage` writes the image stored in the array `image` to *`myfile.hdf`*. If *`myfile.hdf`* exists, the image is appended to the file. If *`myfile.hdf`* does not exist, a new file is created, and the image is written as the first image in the file. The variable `istat` is assigned the value 0 if `DFR8addimage` succeeds; , -1 is assigned if it fails.

for storing and retrieving 8- and 24-bit raster images, palettes, scientific data, and annotations. These interfaces, which are described in detail in chapters 2 through 6, are mutually compatible, and user programs can combine calls to routines in different interfaces when they need to store different kinds of data in the same file.

In some rare cases, an application may require the use of a combination of routines from different interfaces. Just as it is possible to define new HDF tags, it is also possible to build new interfaces by combining routines from two or more existing interfaces.

HDF files tend to be used on several different machines, and HDF interfaces developed at NCSA are implemented on as many machines as possible. An important goal in the development of NCSA HDF user interfaces is to eliminate the necessity of changing program code when moving an application from one machine to another.

HDF Utilities

The HDF *command line utilities* are application programs that can be executed by entering them at the command level, just like other UNIX commands. They make it possible for you to perform, at the command level, common operations on HDF files for which you would normally have to write your own program. For example, the utility `r8tohdf` is a program that takes a raw raster image from a file and stores it in an HDF files in a raster image set.

The HDF utilities provide capabilities for doing things with HDF files that would be very difficult to do under your own program control. For example, the utility `hdfseq` takes a a raster image from an HDF file and displays it immediately on a Sun-3 console.

The HDF utilities are described in detail in Chapter 7, “NCSA HDF Command Line Utilities.”

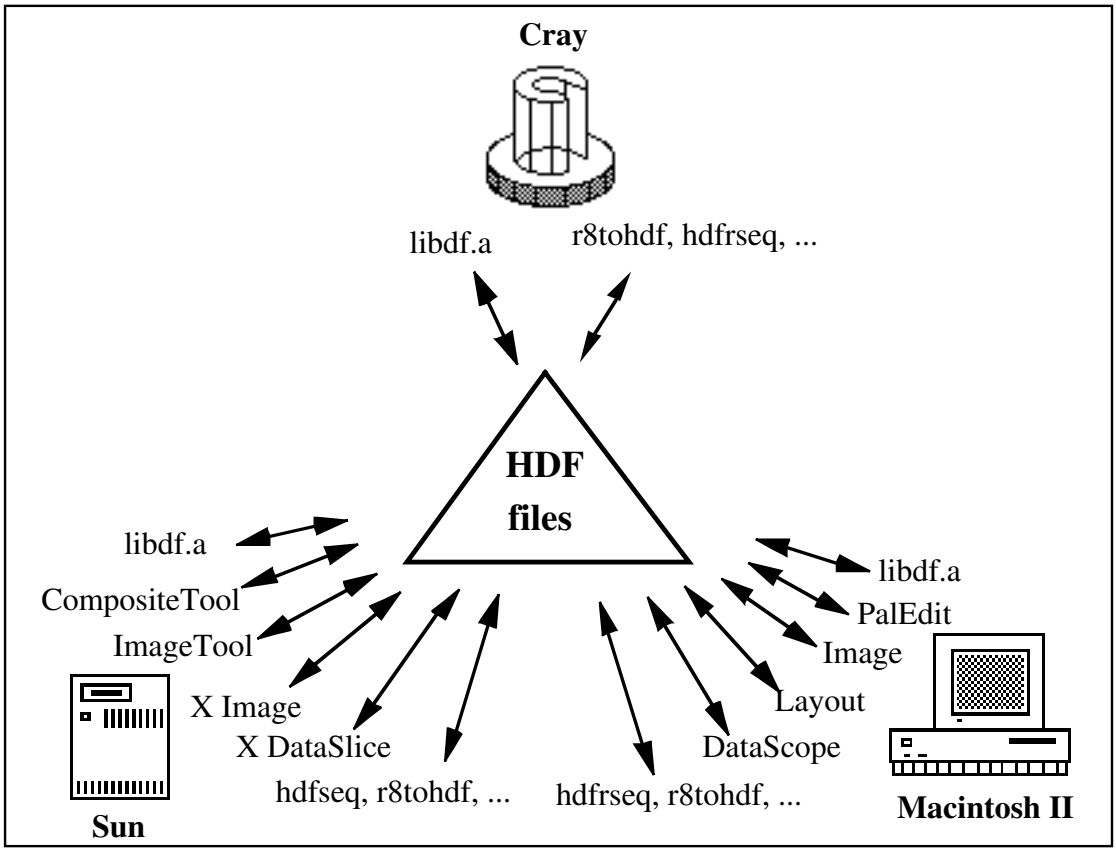
Getting Started with HDF

Appendix D, “Public HDF Directories on NCSA Computers,” contains a list of machines at NCSA that have HDF libraries that are available to all users. If you do not have access to a machine that already has an HDF library, you will need to install it yourself or have your system administrator install it. Although procedures for installing the HDF library vary from one system to another, the basic steps are the same in all cases.

First, you need to get the software from NCSA, as described in the section, “How to Get HDF,” or elsewhere. You might get the actual precompiled library, in which case you could load it into your machine into an appropriate directory. If instead you get the source code for the HDF library, you first have to compile the source code into a linkable library.

Some of the HDF routines are command line utilities, which means that you simply execute them by typing their names in as commands, using the appropriate parameters. Other routines include those that you call from within C or FORTRAN programs. When you write programs that

Figure 1. 3 HDF Software in an Integrated Computing Environment



NCSA Scientific Visualization Software and HDF

The use of HDF files guarantees the interoperability of the scientific visualization tools at NCSA. Some tools operate on raster images, some operate on color palettes, some use images, color palettes, data and annotations, and so forth. HDF provides the range of data types that these tools need, in a format that lets different tools with different data requirements operate on the same files without confusion.

HDF Calling Interfaces

In order to minimize the amount of knowledge you need to have about HDF, calling interfaces are being developed for specific types of applications, such as the storage and display of raster images or scientific data archiving. A *calling interface* is a library of routines that can be called from an application program for storing and retrieving information, including raw data, from a particular type of HDF file.

Different applications typically require different interfaces. Consequently, NCSA HDF provides FORTRAN and C calling interfaces