Chapter 1

Introducing HDF

1.0 Chapter Overview

Welcome to HDF! This chapter will introduce you the very simple, yet fundamental, HDF concepts. It begins by introducing you to HDF with a brief history of its development, follows with a description of HDF and its various layers, and concludes with a discussion outlining some of the benefits of HDF.

If you are already familiar with HDF and how it works, you can skip ahead to Chapter 2 to begin exploring the different HDF data models. If you've worked with HDF in the past but some of the basic concepts are still a bit fuzzy, this chapter will serve as a good review. If are you new to HDF, this chapter will introduce you to the basic capabilities of HDF.

1.1 HDF at a Glance

The HDF project began in 1988 at the National Center for Supercomputing Applications. The HDF development effort originated in response to a need for storing different types of scientific data generated on different kinds of computers. Its primary goal; to help NCSA scientists share their data regardless of its origin.

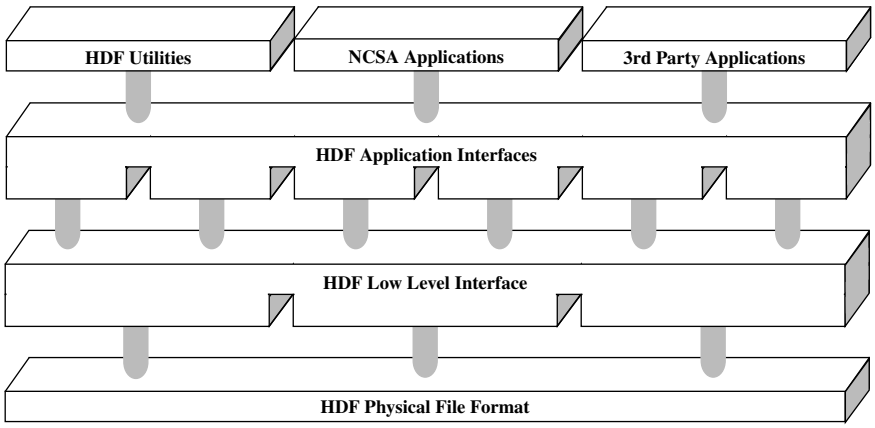
Because HDF is a flexible file format, its use quickly spread beyond NCSA. Commercial vendors, scientists from national laboratories, and many universities began using HDF early in its development cycle. Due to their influence, HDF quickly broadened its scope to include the functionality offered by more popular file formats. As a result, HDF is now used in various computing environments for many different applications.

Although it is gaining popularity, HDF is still only one of many standard file formats available in the scientific community. Achieving interoperability among different standards is still a particularly difficult problem. At first, HDF provided simple utilities for converting files between HDF and other formats. Today however, in addition to adding other features, the HDF development team is exploring new methods for achieving a higher level of interoperability between HDF and other file formats.

1.2 What Is HDF?

Although this question seems simple, it is difficult to answer because it can be addressed in a number of different ways. At its lowest level, HDF is a physical file format. At its highest level, HDF is a collection of utilities and large-scale applications. In between, HDF is a library that provides both a high-level application and a low-level data interface (See FIGURE 1.2a). In practical terms, HDF can be thought of as three interface layers built upon a physical file format.

FIGURE 1.2a **The HDF physical file format supports three levels of interaction.**



Like other file formats, the HDF file format is a predefined method of storing different types of data. The benefit of the HDF file format is that you can read or write different types of data to the same physical file. How you create or access an HDF file is basically up to you, however, it is highly recommended that you use one of the three officially recognized interface layers to do so.

The first interface layer, or the ***low level interface*** is the layer of HDF you'll work with if you wish to work with HDF as a software developer. It contains the source code for things like file I/O, error handling, memory management and physical storage. It's essentially a software tool-kit for skilled programmers who wish to make HDF do something more than what is officially available. If you are a scientist or engineer, it is more likely that you'll be working through the HDF ***application interfaces***.

The HDF application interfaces include six independent modules that are specifically designed to simplify the process of storing and accessing one type of data. If you are primarily interested in reading and writing data to HDF files, you will spend most of your time working with the application interface. Although each interface module requires some programming, all the low level details can be ignored. In most cases, all you have to do is make the right function call at the right time, and the interface will take care of the rest.

At its highest level, HDF includes utilities, NCSA applications, and a variety of other third party applications. Although the HDF utilities are automatically included in the official ***HDF source distribution***, the HDF applications are only available as separate programs. Official NCSA applications and those applications contributed by HDF users are freely available on the NCSA ***ftp server***. In addition, several software vendors also support HDF. Check with your local distributor for more information on commercial packages.

1.3 Why was HDF created?

The HDF file format was created to help today's scientists work in complex computing environments. Unlike the scientific laboratories of yesterday where a single scientist used a single computer to generate and analyze data, today's labs are typically filled with different scientists using many different data sources. These data sources often include scientists at other labs, networked computers, and sophisticated electronic instruments capable of generating vast quantities of data. This growing dependance on multiple data sources introduced the need for a more flexible file format. HDF met this need by allowing scientists to share data files across different computers, access the same data files using different software applications, and store different types of data in the same file.

1.4 Why Use HDF?

Although the world of electronic storage offers many different data formats, there are several reasons why many computational scientists use HDF:

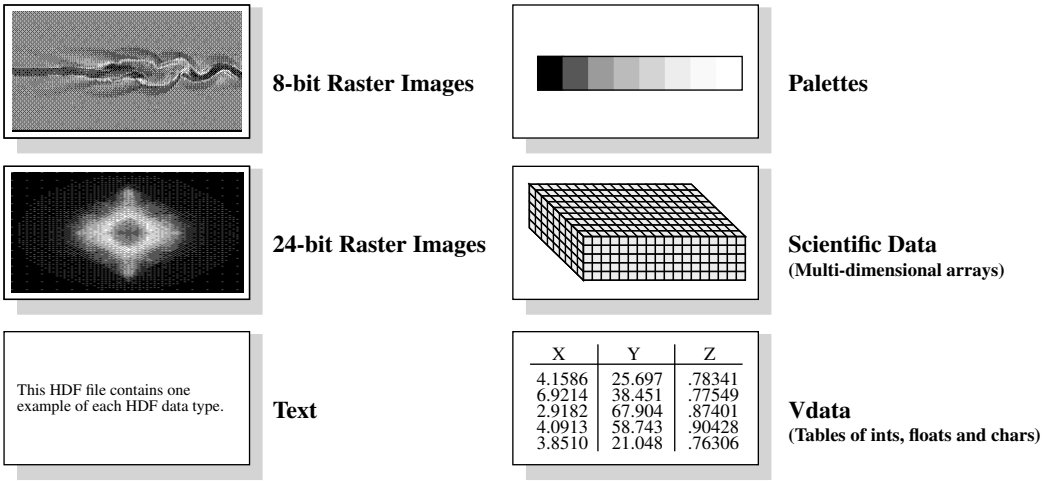
- ***HDF is a versatile file format.*** HDF supports six different data models. Each data model defines a specific type of data and provides a convenient interface for reading, writing, and organizing a unique set of data elements. Because each data model is modularly designed, you need only use the data model appropriate for your immediate needs.
- ***HDF is an extensible file format***. Although HDF currently supports six different data models, there will almost certainly be a need to incorporate new types of data in the near future. HDF is designed to easily accommodate new data models, regardless of whether they are added by the HDF development team or by HDF users.
- ***HDF is self-describing.*** Inside every HDF file there is a data index which identifies and describes the contents of the file. This makes HDF a convenient file format for sharing data because everything you (or the scientists you are working with) need to know about the data can be stored in the file along with the data. HDF eliminates the need for scribbled reminders describing how and what data is stored in a file.
- ***HDF is a flexible file format.*** With HDF, you can group sets of related data together and then access them as a group or as individual objects. You can even group existing sets together into larger sets and still access each subset individually. As you work with HDF, you will find that it can make storing and organizing data extremely convenient.
- ***HDF is a portable file format.*** HDF files are portable in the sense that they can be shared across platforms. For example, an HDF file created on one computer (a Cray supercomputer for example) can be used on another system (a Macintosh, perhaps) without modification.
- ***HDF is a portable library.*** Once the HDF library is compiled for your machine, its functions can be called from inside your FORTRAN or C programs. By using the HDF library, most of the programming necessary for reading or writing HDF files is already done for you.
- ***HDF is available in the public domain.***

HDF is a versatile file format.

If you are like most scientists, it is fairly likely that you will encounter many different types of data. This data often includes programs and their output, data collected from the real world, and data received from other scientists. In order to keep all this information organized, you'll need some way of storing and organizing different types of data. With HDF, you can store a variety of data types in the same file (See FIGURE 1.4a). Once the data is in a file, you can add meaningful annotations that describe the origin, or other characteristics of the data.

To demonstrate the versatility of HDF, consider an example. Assume an HDF file contains a program and its floating-point output. Suppose you wish to view the data as a series of images in an animation sequence. The first step toward an animation is to convert the floating point data into 8-bit integers. Once the data is converted, you must create and order the series of images. Finally, you must assign an appropriate color-look up table. Ordinarily the results of each step are stored in separate files, however, with HDF, the source code, its floating point output, the 8-bit images, and color look-up table can all be stored in the same file. As you can see, HDF is truly versatile in the sense that it can store any number (limited by available memory, of course) and variety of data types in the same file.

FIGURE 1.4a **HDF can stores several different types of data in the same file.**



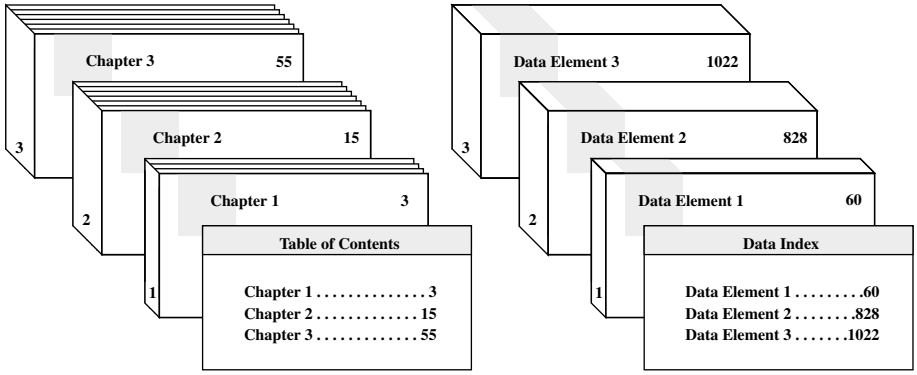
HDF is an extensible file format.

HDF is specifically designed to easily incorporate new data models. Even though there are already a great number of recognized data types, there will almost certainly be a need for new ones. The HDF development team is already considering the best way to create new data types for things like voice and video. If you need to work with a data type that HDF does not currently support, it is possible to create your own. However, only those people who have access to your code will recognize your new data type. You can learn more about creating new data types in the HDF Users Guide.

HDF files are self-describing.

HDF files are considered self-describing because inside every file is a list describing the contents of the file. Perhaps the best way to think of an HDF file is to think of a multi-chapter book with a table of contents (See FIGURE 1.4b). HDF files are “data books” in the sense that each chapter contains a different type of data or ***data element***. Just as a book uses a table of contents to list its chapters, an HDF file uses a “data index” to list its data elements. Consequently, the “table of contents” for an HDF file can be though of as a list of all the data elements contained in the file. Although keeping an active data index may seem a bit complicated, in most cases HDF will maintain control over how and where data elements are added. Your job is simply to identify what data elements you need to add to which files.

FIGURE 1.4b **The building blocks of an HDF file are similar to those found in a conventional book.**

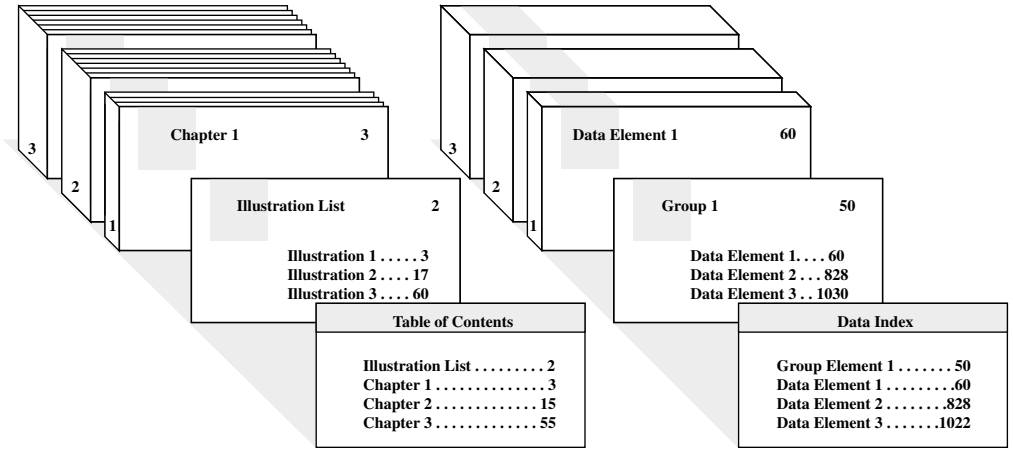


HDF is a flexible file format.

What makes HDF a flexible file format is its ability to access data elements regardless of how they are stored. With HDF, you can store data elements one-to-a-file, add multiple data elements to a file, or group related data elements together in the same file. The level of organization required for a particular collection of data is typically determined by the nature of the data, and your level of familiarity with HDF.

To better understand how data elements are grouped in HDF files, recall that every HDF file can be thought of as a multi-chapter book with a table of contents. The process of grouping data elements is similar to the process of adding a list of related items to a book. Say for example, you wish to add a list of illustrations to a book. First you must identify the illustrations you wish to include, then you must compile the list and insert it into the book. Finally, you must create an entry for the list in the table of contents. HDF files perform similar tasks when asked to group data elements. First the group members are identified, then they are compiled into a list and added to the file. Finally, an appropriate entry for the group is added to the data index (See FIGURE 1.4c on page 6).

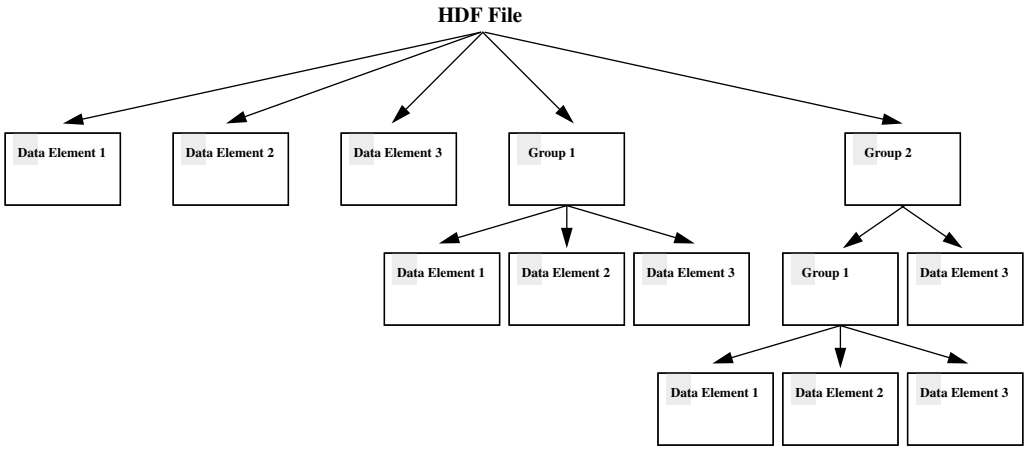
FIGURE 1.4c **Grouping a set of data elements in an HDF file is similar to adding a list to a book.**



Once a ***group element*** appears in the data index, HDF treats it just like an individual data element. In other words, groups can be added to other groups just as if they were single data elements. For example, grouping Data Element 3 with Group Element 1 will form a new group and add another group reference to the data index. The second group will not interfere with the first group or any of the other data elements contained in the file. It will simply add another group to the file.

By intelligently grouping data elements, you can create extremely complex or very simple organizational structures. Conceptually speaking, each data group represents a node in a data tree. Consider the two group elements described in the previous example. The first group contained three individual elements, and the second contained one group and one individual element. This particular arrangement translates into a three-tier data tree with each group branching into the next lowest tier (See FIGURE 1.4d).

FIGURE 1.4d **The organizational structure of an HDF file can be represented by a data tree..**



As you work with HDF, you will discover that HDF will most likely create the appropriate data structures for you. However, for more complex structures you may be called upon to create your own. You can learn more about creating data hierarchies in the HDF User's Guide.

HDF is a portable file format.

The official ***HDF distribution*** contains all the source code you will need to create an HDF library for your machine. The HDF distribution provides C code for a variety of different computers. At this point, check the list below to see if your machine is officially supported.

Convex (UNIX)	**IBM PC** (MS-DOS/Windows)	**NeXT** (NeXTStep)
Cray X-MP/2 (UNICOS)	**IBM RT** (AIX)	**Silicon Graphics** (Irix)
DecStation (Ultrix)	**Macintosh** (MacOS)	**Sun 3, Sun 386, Sparc** (SunOS)
HP 9000 (HPUX)	**IBM RS6000** (AIX)	**Vax** (VMS, Ultrix)

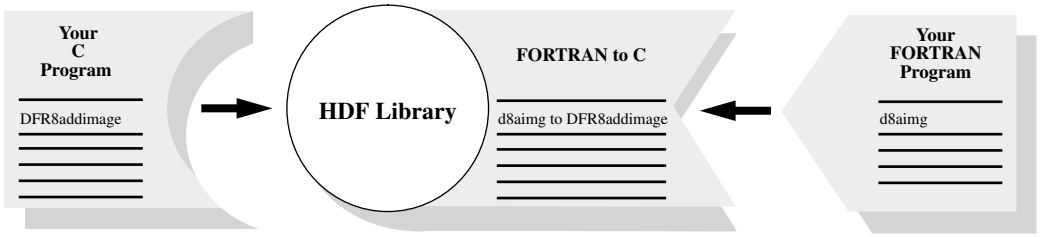
If your computer does not appear in this list, there is still some hope. NCSA may have ported HDF to your machine since the publication of this document. There is also a chance that a user has contributed code for your machine. Although contributed code is not included in the official release, it is available on the NCSA ftp server in the HDF/contrib directory. Check the current HDF README for more details.

HDF is a FORTRAN and C callable library.

The ***HDF library***, like other software libraries, provides object code for a collection of frequently used functions. Unlike other software libraries however, the HDF library is accessible from both C and FORTRAN programs. By using its own FORTRAN to C converter, the HDF library can support HDF function calls made from both C and FORTRAN programs.

Even though the HDF library is accessible to both C and FORTRAN programmers, some FORTRAN compilers, such as UNICOS FORTRAN (CFT77) on the CRAY-2 system, only accept function names that are eight or fewer characters. To avoid compiler errors, HDF provides two names for each function; one for use with the C library, and a shorter version for use with the FORTRAN library. For example, **d8aimg** is the FORTRAN equivalent for **DFR8addimage**. Both function calls will add an 8-bit raster image to an HDF file (See FIGURE 1.4e).

FIGURE 1.4e **HDF uses a function call converter to interface FORTRAN calls to its C library.**



Before you can use ***HDF function calls*** in your C or FORTRAN programs, the HDF library must be installed on the machine you plan to use. If at some point in the past HDF was compiled for your machine, all you need to do is check to see that it is current. To do so, you can ask your system administrator or check its ***version number*** and compare it with the current version available on the ftp server. If you do not have access to a machine with a usable HDF library, you (or your system administrator) will need to install it before you attempt to link your code to the HDF library. Complete installation instructions are available on the NCSA ftp server.

HDF is available in the public domain.

HDF source code is made available in the public domain. Specifically, NCSA gives to the public domain all rights for future licensing, resale, and publication. NCSA asks, but does not require, that the following message be included in all derived works:

> ***Portions developed at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, in collaboration with the Information Technology Institute of Singapore.***

If you want to see more software like HDF, please send us a letter (email or U.S. mail), telling us what you are doing with HDF. We'd like to know what you are working on, so please include an abstract or short description of your work. We'd also like to know how HDF has helped you, for example, by increasing your productivity or allowing you to do things you could not do before. A bibliography of your work would be extremely helpful.

We encourage you to cite the use of HDF in your publications. This is a new kind of shareware. You share your science and successes with us, and we can get resources to share more software like HDF with you. Please send your letters to:

U.S. mail:

NCSA Software Tools Group, HDF
152 Computing Applications Building
605 E. Springfield Ave.
Champaign, IL 61820

Electronic mail:

softdev@ncsa.uiuc.edu
softdev@ncsavms.bitnet

Although we try hard to test our code, errors are an unavoidable part of software development. If you encounter difficulty with HDF and you are convinced that it is a bug, feel free to report it to us at the addresses listed above. If you are experiencing difficulty and you are not sure what is causing the problem, you can get help by contacting the HDF support staff at NCSA. Our support staff will try to respond as quickly as possible. If they are unable to respond immediately, please be patient.