Multiple References and File Compaction

Multiple references to a single data element are quite common. The general purpose routine DFdup (see Chapter 2) generates a new reference to data that is already pointed to by another DD. If DFdup is used several times, there could be several DDs that point to the same data element. It is important to note that when such a multiply referenced data element is moved, the various DDs that pointed to the data element before it was moved are *not* automatically adjusted to point to the data element in its new location.

For example, when DFaccess is called for write access, the referenced data is copied to the end of the HDF file to allow the user to append to that element. The original data is not moved, only copied. If there are multiple references to that data, the old references still point to the old data.

Consequently, compaction of a file should be done in the following manner. Proceed through the data descriptors in order. For each descriptor, determine whether the data has already been copied; if not, copy the data into a new file and update its descriptor. If the data has already been copied, as in the case of a multiple reference, then the descriptor should be placed in the new file as a multiple reference. Any data that no longer has a DD reference should not be copied to the new file.

After updating elements, you may need to call DFdescriptors again to make sure that your copy of the descriptors is up-to-date.

Of course, for most programs you will be writing, you only need to create a completely new file, add elements to an existing file, or read a pre-existing file. For all of these operations, consistency is completely handled by the lower level HDF interface, even in the case of multiple references to the same data.

Using Reference Numbers to Organize Data Objects

Reference numbers distinguish between one data object and others with the same tag. It is often tempting to assign meaning to reference numbers beyond this distinction. Such uses of reference numbers are generally discouraged because most HDF software assumes that reference numbers can be chosen or assigned in any order, as long as all tag/ref combinations in a file remain unique.

On the other hand, in some situations reference number can be used easily and effectively to add meaning to a file—for instance, to provide informal grouping and ordering of data objects. For example, suppose you want to create a movie out of the images in an HDF file. The images need to be ordered in some way and each image needs to be associated with other data objects. In such cases, you should refer to the following rules to assign reference numbers, especially among sets of tags that are required to go together.

1. Assign reference numbers in increasing order.

2. When a companion object is required (for example, image dimension is needed for images), use the object with the current reference number. If none exists, use the object with the next lowest reference number.

3. When there are several combinations, cycle through reference numbers for each tag type in ascending order in all combinations that do not break the order.

For example, suppose the following data identifiers are in a file: IP8[1], ID8[1], RI8[1], RI8[2], IP8[2], IP8[3], IP8[4], RI8[4], RI8[10], DIL[1], DIL[10]. Objects in the file should be grouped in the order shown in Table 4.1.

Table 4.1 Sample Grouping of Data Objects in an HDF File

IP8	ID8	RI8	DIL	
1	1	1	1	palette[1], dimensions[1], image[1], label[1]
2	1	2	1	change to palette[2] and image[2]
3	1	2	1	change to palette[3]
4	1	4	1	change to palette[4] and image[4]
4	1	10	10	change to image[10] and label[10]

Of course, these conventions only work if all of the programs that touch a file do so without altering the reference numbering scheme. If this cannot be guaranteed, you have a second solution—to impose order on the file by means of a sorted, keyed index. This is a good solution because it can provide random access as well as ordered access to data; however, it adds a level of complexity to what might otherwise be a simple sequential file organization.

Chapter Overview

The specification of HDF described in the previous chapter is not sufficient to guarantee its success. With a general structure such as HDF to work with, you may be tempted to create overly clever, obscure, and complex files, defeating the purpose behind the design of HDF—to facilitate the interchange of data among differing applications, programs, and computer systems. To guard against these problems, a number of guidelines on the use of HDF are provided: some are implicit in the discussions in other sections of this document and others are presented in the NCSA HDF manual. Additional guidelines not covered elsewhere are introduced in this section.

Byte Order

The byte-ordering of data stored in memory and on files varies from machine to machine. It is important to address this possibility in any software that processes data from files that might have visited more than one machine.

In most cases, the existence of "byte-swapped" data can be detected by examining the Machine Type (MT) tag, which is described in Appendix A. Unfortunately, a program cannot look at the MT until it has opened an HDF file and looked at the header, so it is important that the byte order of the header be the same, no matter what machine is used to create an HDF file. Therefore, to maintain machine portability when developing software for machines that swap bytes, you should make sure that the characters are read and written in the exact order specified (i.e., ^N^C^S^A).

Naming and Assigning Tags

Tags that are to be made available to a general population of HDF users should be assigned and controlled by NCSA. Tags of this type are given numbers in the range 8,000-15,999. If you have an application that fits this criterion, contact NCSA at the address listed on the README page at the beginning of this manual and specify the tags you would like. For each tag, your specifications should include a suggested name, information about the type and structure of the data that the tag will refer to, and information about how the tag will be used. Your specifications should be similar to those contained in Appendix A. NCSA will assign you a set of tags for your application and include your tag descriptions in its documentation.

Tags in the range 16,000-32,000 are user-definable. That is, you can assign them for any private application. Of course, if you use tags in this range you need to be aware that they may conflict with other people's private tags.

Chapter 4 HDF Conventions

Chapter Overview

Byte Order

Naming and Assigning Tags

Using Reference Numbers to Organize Data Objects

Multiple References and File Compaction