Ada 95 Bindings for the NCSA Hierarchical Data Format

Bruce R. Barkstrom
Atmospheric Sciences
NASA Langley Research Center
Hampton, VA 23681-2199
1-757-864-5676
b.r.barkstrom@larc.nasa.gov

1. ABSTRACT

This paper describes Ada95 bindings for HDF4 and HDF5, the current versions of the NCSA Hierarchical Data Format (HDF). These self-describing file formats are intended for storage of large, diverse collections of scientific data and for retrieving subsets of these data. The libraries also support data compression, chunking of large arrays, and automatic conversion of vendor-specific binary formats for a variety of data types.

1.1 Keywords

File formats, Hierarchical Data Format, HDF, HDF4, HDF5, Self-documenting files

2. INTRODUCTION

While the four standard file I/O packages that are part of the Ada95 language cover many I/O needs, only the Streaming I/O package deals with storage and retrieval of heterogeneous data types. A programmer who needs to store such data is still faced with the rather formidible task of defining the order of the data structures and procedure calls – and even after considerable work is likely to have an idiosyncratic file format that may be difficult to transfer from one platform to another.

Fortunately, the ability of Ada95 to easily interface with other languages has provided an opportunity for this language to obtain bindings to the Hierarchical Data Format (HDF) libraries developed by the HDF Group at the National Computational Sciences Alliance (NCSA). This family of file formats appeared about the time the early Mosaic browser appeared and has been evolving since.

HDF files are intended to provide self-documenting storage of scientific data. This means that HDF provides structures that allow the file format to contain data about the file structure and descriptive information about the data contained in the file. By reading an appropriate sequence of data structures in the file, a data user can extract the information needed to understand the kind of data in the file, the size, shape, and data type of the arrays in the file, and even documentation about the file contents. The data types supported by the HDF files include strings, integers, and floating point values, as well as pointers and other linking elements. As a result, the Ada95 bindings to HDF4 and HDF5 offer an opportunity for the Ada community to freely avail themselves of very powerful tools for storing and retrieving heterogeneous data structures (and even procedures), as well as visualization tools and other helps at organizing and using data.

This paper provides a brief overview of the HDF file formats themselves, followed by a description of the Ada95 bindings and their documentation.

3. HDF

As the HDF4 Reference Manual [1] notes "Scientists commonly generate and process data files on several different machines, use various software packages to process files and share data files with others who use different machines and software. Also, they may include different kinds of information within one particular file, or within a group of files, and the mixture of these different kinds of information may vary from one file to another. Files may be conceptually related but physically separated. For example, some data may be dispersed among different files and some in program code. It is also possible that data may be related only in the scientist's conception of the data; no physical relationship may exist." The latter conception may be particularly important where scientists have placed data in an array whose

SIGAda 2001 09/01 Bloomington, MN, USA
© 2001 ACM 1-58113-392-8/01/0009...$5.00

underlying structure corresponds to sampling a physical field with an implicitly defined grid which is widely understood in that scientific discipline.

The Reference Manual continues "HDF addresses these problems by providing a general-purpose file structure that:

- Provides the mechanism for programs to obtain information about the data in a file from within the file, rather than from another source.
- Lets the user store mixtures of data from different sources into a single file as well as store the data and its related information in separate files, even when the files are processed by the same application program.
- Standardizes the formats and descriptions of many types of commonly-used data sets, such as raster images and multidimensional arrays.
- Encourages the use of a common data format by all machines and programs that produce files containing specific data.
- Can be adapted to accommodate virtually any kind of data."

4. HDF4

HDF emerged at about the time the Mosaic web browser appeared. At that time, its creators viewed HDF as a way of providing platform-independent file structures with a standard set of browse tools that would allow diverse groups of scientists to view the data in files and to document their contents. Over time, the various scientific groups working with NCSA requested additions to the data structures and procedures included in HDF.

The fourth version of the Hierarchical Data Format, or HDF4, includes several basic data structures, such as specialized arrays for raster images with various numbers of bits per pixel, together with their associated palettes. More general arrays are describable as "Scientific Data Sets," that have annotation fields. Data stored in tables, such as the data one might find in relational databases can be placed in data structures identified as "Vdata" that can be organized into "Vgroups."

It is particularly important to note that HDF files are self-describing. This means that, for each HDF data structure in a file, there is comprehensive information about the data and its location in the file. This information is often referred to as metadata.

The Scientific Data Sets and Vdata structures can include many types of data. For example, it is possible to store symbolic, numerical and graphical data within a single HDF4 file by using appropriate HDF data structures.

If we move beyond the data structures inside the files, we note that HDF4 can be viewed as several interacting layers of software. At the lowest level, HDF4 is a physical file format for storing scientific data. At the highest level, HDF4 is a collection of utilities and applications for manipulating, viewing, and analyzing data stored in HDF4 files. Between these levels, HDF4 is a software library that provides high-level and low-level programming interfaces. It also includes supporting software that make it easy to store, retrieve, visualize, analyze, and manage data in HDF files.

The basic interface layer, or low-level API, is intended for software developers. It was designed for direct file I/O of data streams, error handling, memory management, and physical storage. Thus, this layer provides a software toolkit for experienced programmers who wish to make HDF4 do something more than what is currently available through the higher-level interfaces. For most purposes, these low-level routines are available only in C.

The HDF4 application programming interfaces, or APIs, include several independent sets of routines, with each set specifically designed to simplify the process of storing and accessing one type of data. Although each interface requires programming, all the low-level details can be ignored. In most cases, all one must do is make the correct function call at the correct time, and the interface will take care of the rest. Most HDF interface routines are available in both FORTRAN-77 and C.

The Ada95 bindings for HDF4 provide an alternative interface. These bindings provide direct equivalents to the C interfaces containing procedure and function calls, as well as the HDF4 data structures.

HDF4 is very well documented [1]. The documentation available on-line includes a ***User Guide*** and a ***Reference Manual***. Furthermore, the HDF web site also provides a tutorial for the API's, so that programmers can systematically develop proficiency in efficiently using this file format. The Ada95 bindings we provide also include procedures equivalent to the C programs in the HDF4 documentation.

5. HDF5

HDF4 uses a "linked" representation of the data structures a file contains. When HDF4 was designed, the "reference" and "tag" structures that provide the links were limited to

32-bit referencing. As a result, the physical size of an HDF4 file is constrained to 2 gigabytes.

In 1998, the HDF Group released a beta version of a new file structure that was intended to substantially improve on HDF4. As the HDF5 Introduction [2] notes: "The development of HDF5 is motivated by a number of limitations in the older HDF format and library. Some of these limitations are:

- A single file cannot store more than 20,000 complex objects, and a single file cannot be larger than 2 gigabytes.
- The data models are less consistent than they should be, there are more object types than necessary, and datatypes are too restricted.
- The library source is old and overly complex, does not support parallel I/O effectively, and is difficult to use in threaded applications.

HDF5 includes the following improvements.

- A new file format designed to address some of the deficiencies of HDF4.x, particularly the need to store larger files and more objects per file.
- A simpler, more comprehensive data model that includes only two basic structures: a multidimensional array of record structures, and a grouping structure.
- A simpler, better-engineered library and API, with improved support for parallel I/O, threads, and other requirements imposed by modern systems and applications."

The newer and much simpler data structure for HDF5 places arrays within "Datasets," which are then placed in "Groups." Rather than providing distinct "image" types, HDF5 recognizes that images are subsets of multi-dimensional arrays. Thus, there is no point in introducing additional, distinct subtypes of arrays. The file is self-describing because datasets that include arrays also must include a "Dataspace" that quantifies the number, indexing order, and size of the array. Datasets must also include a "Datatype" structure that quantifies the individual element types in the array. HDF5 is particularly interesting because it allows both simple data types (bytes, integegers, floats, characters, and strings) and compound types that are equivalent to Ada95 records.

As with HDF4, the HDF5 API provides routines for creating HDF5 files, creating and writing groups, datasets, and their attributes to HDF5 files, and reading groups, datasets and their attributes from HDF5 files. It is particularly noteworthy that HDF5 provides several functions that specifically support the following features:

- Data compression
- Chunking, in which portions of the array are accessible in "chunks" that are efficient data structures for accessing smaller portions of the file than hyperslabs
- Irregularly-shaped array subset extraction

HDF5 not only comes with a library, documentation, and a tutorial, it also has several tools that make life easier for developers using this file format. For example, there is a file content "dumper" that provides a printout of the data structures in the file. HDF5 also has an XML description that provides a useful formalism for designing efficient data structures.

6. THE HDF ADA BINDINGS

HDF is becoming widely used in scientific data holdings, such as those of NASA's Earth Observing System (EOS) Data and Information System – EOSDIS. At present, EOSDIS is adding about one TB per day to NASA's Earth Science data holdings. In the near future, this rate will increase to about three TB per day. The EOS Project has directed NASA Principal Investigators to use HDF for the data format for all EOS missions. Thus, this data format is very important to the author (he is Instrument Principal Investigator for the investigation of Clouds and the Earth's Radiant Energy System: CERES), as well as many of his colleagues.

Because CERES uses Ada for some of its scientific data production, the author felt that it would be useful to provide Ada95 bindings for the HDF5 and HDF4 libraries. He contacted John S. Walker, who had provided Microsoft Foundation Class bindings for Aonix, Inc., to translate the C and C++ header files into Ada95 bindings. The author tested the bindings by using them to create Ada95 programs that duplicate the functions of the HDF tutorials available through the HDF web site [1][2], using the Aonix ObjectAda compiler and Integrated Development Environment on Windows NT machines.

As one might expect, the bindings include a number of data structure definitions and procedure interfaces. A simple program to create an HDF5 file uses the normal Ada "with" statements to include the bindings and then calls the library procedures using the interfaces, as Listing 1 shows.

```
with VC;            use VC;
with HDF5Constants; use HDF5Constants;
with HDF5C;         use HDF5C;
with interfaces.c.strings;

procedure H5_Crtfile is
  use type interfaces.c.strings.chars_ptr;
  fn : interfaces.c.strings.chars_ptr
     := Interfaces.c.strings.new_string(
         str => "file.h5");
  flags : VC.Unsigned
       := HDF5Constants.H5F_ACC_TRUNC;
  create_plist : HDF5Constants.hid_t
              := HDF5Constants.H5P_DEFAULT;
  access_plist : HDF5Constants.hid_t
            := HDF5Constants.H5P_DEFAULT;
  File_Id      : HDF5Constants.Hid_T;
   -- file identifier --
  Status       : HDF5C.Herr_T;
begin
  -- Create a new file using default properties.
  File_Id
    := H5Fcreate(filename     => fn,
                 flags        => flags,
                 create_plist=> create_plist,
                 access_plist => access_plist;
  -- Terminate access to the file.
  Status := H5fclose(File_Id);
end H5_Crtfile;
```

Listing 1. Ada95 Program to Create a Simple HDF5 File Using the HDF Bindings.

The author expects to provide the Ada bindings to interested members of the community, probably as contributed software to the HDF Group at NCSA. The powerful and standardized data access that HDF makes possible appears to offer a number of interesting prospects, particularly for distributed databases that are capable of selecting subsets of large, scientific files and formatting them into much smaller packages that are suitable for transmission over the Internet.

In addition to the binding source code, the published Ada Bindings are expected to be supported by a tutorial that will include not only the material equivalent to the standard API documentation provided by the HDF Group, but also a Windows GUI form of an HDF editor (written entirely in Ada95) that will allow users to create or access and edit HDF data structures for both HDF4 and HDF5.

7. ACKNOWLEDGMENTS

This work has been supported by NASA's CERES investigation. The author is very appreciative of the interest and encouragement of the HDF Group and its leader, Dr. Michael Folk. The HDF Ada95 bindings have been provided by John Walker, using his technology to automatically convert the C and C++ header files into useful bindings. Without this technology, these bindings would not have been developed. The author is also extremely grateful to the support provided by Aonix, Inc. In addition to the excellent support provided by the ObjectAda compiler and its IDE, the author is particularly appreciative of the support of Greg Gicca and of Lyn Henderson, without which it would not have been possible to have developed these bindings.

8. REFERENCES

[1] HDF Group. HDF[4] User's Guide, available at http://hdf.ncsa.uiuc.edu/.

[2] HDF Group. Introduction to HDF5 Release 1.2, available at http://hdf.ncsa.uiuc.edu/