HDF4.3 使用简介

目 录

第一章 HDF 介绍

1.1 本章概况

本章将解释什么是分层数据格式（HDF）和创建它的意义，在进一步将介绍 HDF 文件基本格式的基础上，还将介绍如何使用 HDF 文件的初始数据结构及其方法。

1.2 什么是 HDF

HDF 是用于存储和分发科学数据的一种自我描述、多对象文件格式。HDF 是由美国国家超级计算应用中心（NCSA）创建的，以满足不同群体的科学家在不同工程项目领域之需要。HDF 可以表示出科学数据存储和分布的许多必要条件。HDF 被设计为：

- ✧ 自述性：对于一个 HDF 文件里的每一个数据对象，有关于该数据的综合信息（元数据）。在没有任何外部信息的情况下，HDF 允许应用程序解释 HDF 文件的结构和内容。
- ✧ 通用性：许多数据类型都可以被嵌入在一个 HDF 文件里。例如，通过使用合适的 HDF 数据结构，符号、数字和图形数据可以同时存储在一个 HDF 文件里。
- ✧ 灵活性：HDF 允许用户把相关的数据对象组合在一起，放到一个分层结构中，向数据对象添加描述和标签。它还允许用户把科学数据放到多个 HDF 文件里。
- ✧ 扩展性：HDF 极易容纳将来新增加的数据模式，容易与其他标准格式兼容。
- ✧ 跨平台性：HDF 是一个与平台无关的文件格式。HDF 文件无需任何转换就可以在不同平台上使用。

1.3 为什么创建 HDF

科学家通常在不同的机器上生成和处理数据文件。各式各样的软件包被用来多种处理文件，同时也与其他使用不同机器和软件的人共享数据文件。在一组文件里，这些文件也许包含不同类型的信息。这些不同类型的信息混合结构在一个文件里的意义与在另一个文件里的意义不同。这些文件也许概念上有关但在实质上却不同。例如，一些数据（符号、数字和图形）也许在不同文件中被分开，但在程序中却是在一起。有些数据也许在科学家的概念中是相关的，但实际上并不存在物理联系。HDF 通过提供一个一般目的的文件结构来表明这些问题：

- ✧ HDF 为程序提供一个从数据文件本身获取数据（数据元）信息的机制，而不是其他来源。
- ✧ 使用户把不同来源的混合数据存放在一个文件里，同时也可以把数据和与之相关的信息存放在不同的文件里，即便是用同一个应用程序也可以处理这些文件。
- ✧ 常用的许多类型数据集，如光栅图像和多维数组，对其格式和描述实行标准化。
- ✧ 支持使用标准的数据格式，因为所有的机器和程序都会生成一个有明确含义的数据文件。HDF 实质上能被用于任何类型的数据。

1.4 HDF 的 6 种基本数据类型

HDF 提供 6 种基本数据类型：光栅图像（Raster Image），调色板（Palette），科学数据集（Scientific Data Set），注解（Annotation），虚拟数据（Vdata）和虚拟组（Vgroup）。所有的这 6 种基本数据类型由图 1a 来阐明。

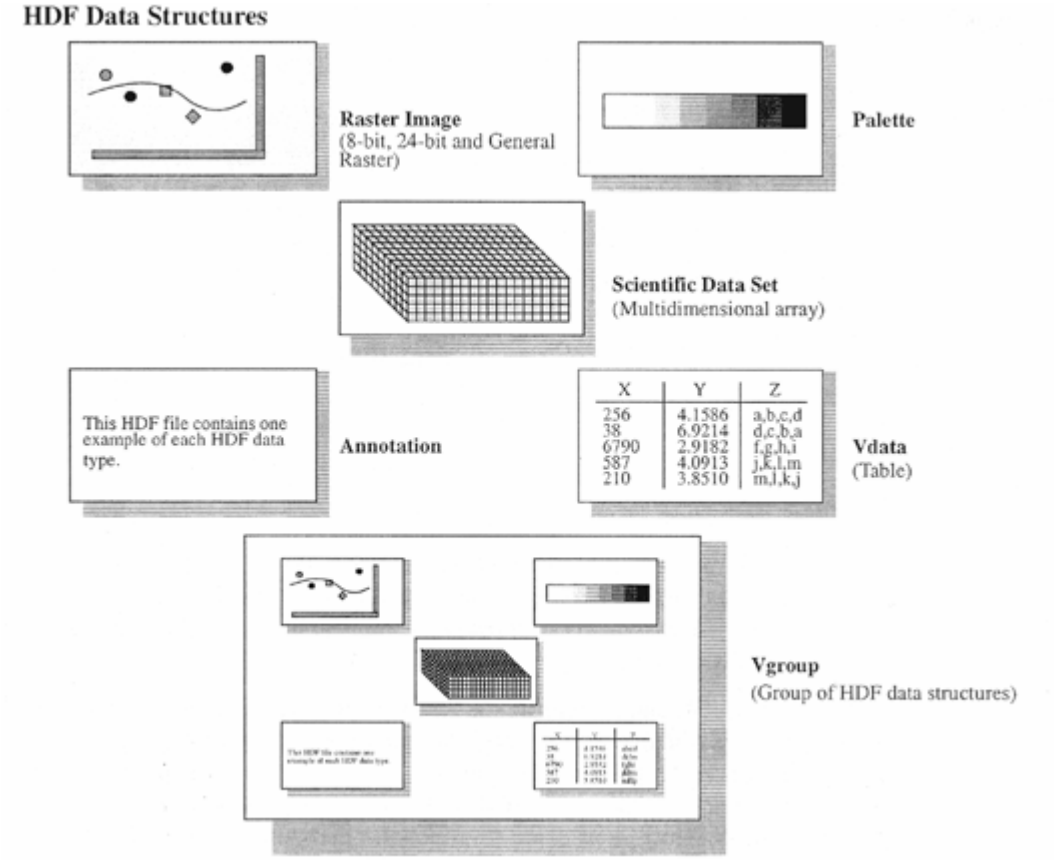


图 1a HDF 数据类型

- ✧ Raster Image 数据模型被设计成能为光栅图像数据的存储和描述提供一个灵活的方法，包括 8 比特和 24 比特光栅图像。详细介绍参见第 3 章。
- ✧ Palette 作为颜色查找表提供图像的色谱。它是一个表格，其表中每列的数字表示特定颜色的数字。细节参见第 4 章。
- ✧ Scientific Data Set 模型是用来存储和描述科学数据的多维数组。详见第 5 章。
- ✧ Vdata 模式是用来存储和描述数据表格的结构。详见第 6 章。
- ✧ HDF 的 Annotations 是文本字符串，用来描述 HDF 文件，或 HDF 文件包含的 HDF 数据对象。详见第 7 章。
- ✧ Vgroup 结构模型被设计为与相关数据对象有关。一个 Vgroup 可以包含另一个 Vgroup 以及数据对象。任何 HDF 对象都可以包含在一个 Vgroup 中。详见第 8 章。

HDF 库为每一个数据模型提供一个应用编程接口。每一个数据模型和相应的应用编程接口将在下一章详细描述。

1.5 HDF 文件的 3 层交互

HDF 文件可以在几个交互层次中可视。在最底层，HDF 是一个存储科学数据的物理文件格式。在它的最高层，HDF 是集工具和应用于一体的数据文件，可以对 HDF 文件中的数据进行修改、显示和分析。在这两个层次之间，HDF 是一个能提供高层和底层编程接口的软件库。图 1b 为这些接口层的示意图。

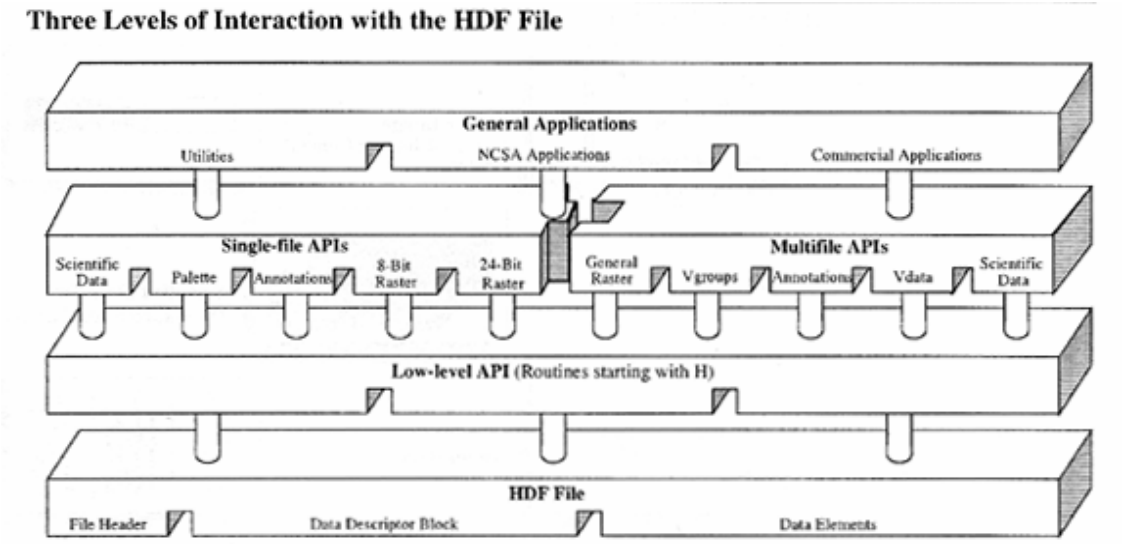


图 1b HDF 文件的 3 层交互

普通应用接口层，或称为底层的应用编程接口（API），是为软件开发者保留的。它是为数据流的直接文件 I/O、错误处理、内存管理和物理存储而设计的。它是一个为有经验的 HDF 程序员提供的软件工具。比较目前从高层接口得到的功能，通过使用这些基本接口层，HDF 程序员创建 HDF 文件时能够做更多的事。底层接口例程只提供 C 语言。

HDF APIs（HDF 应用编程接口）分为两类：多文档接口（用于新版本）和单文档接口（用于旧版本）。多文档接口是提供从一个应用中同时连接几个 HDF 文件的接口，这点很重要，但单文档接口并不支持这点。用户在开发新的接口和界面时，必须提醒他们是在一个改进了的新接口版本下开发的。为了向上兼容，旧版本仍然保留。

HDF APIs 包含几个独立的例程集，每个例程集是专门为简化一种数据类型的存储处理而设计的。这些接口在图 1d 中作为单文件和多文件层。尽管每个接口都要求程序调用，但所有底层细节都可以忽略。大多数情况下，只须在正确的时间调用正确的函数，剩下的事就由接口程序处理。多数 HDF 接口例程都有 FORTRAN-77 和 C 语言。也有用于 Java 程序员访问 HDF 文件的 Java HDF 接口程序。

1.6 HDF 文件格式

最好的办法是把 HDF 文件看成为一本有表格内容的多章节书。HDF 文件是“数据书”，其中每章都包含一个不同类型的数据内容。正如书籍用一个目录表列出它的章节一样，HDF 文件用“data index”（数据索引）列出其数据内容。

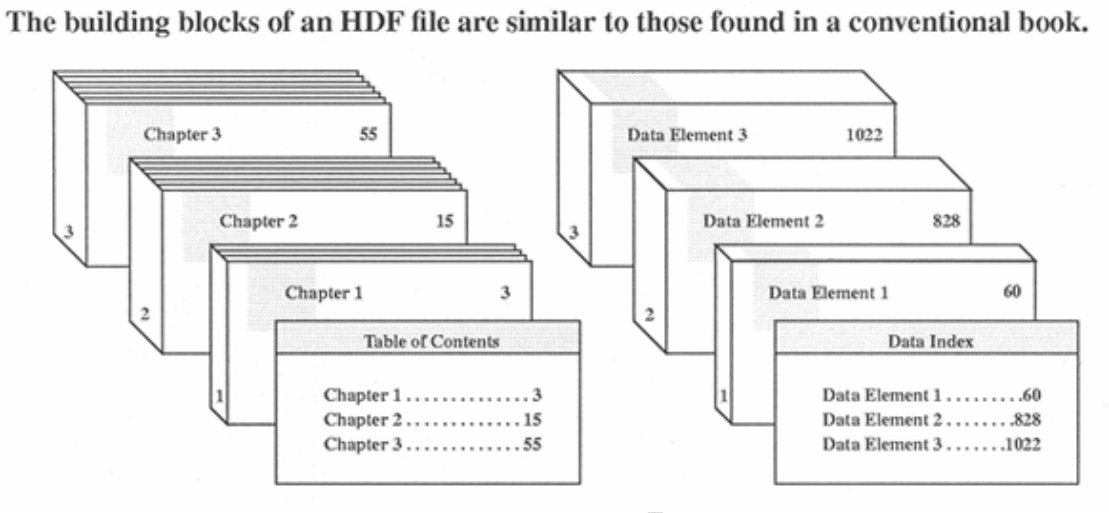


图 1c　HDF 文件结构同传统意义上的多章节书的比较

如图 1d，HDF 文件结构包括一个 file id（文件号）、至少一个 data descriptor（数据描述符）、没有或多个 data element（数据内容）数据内容。

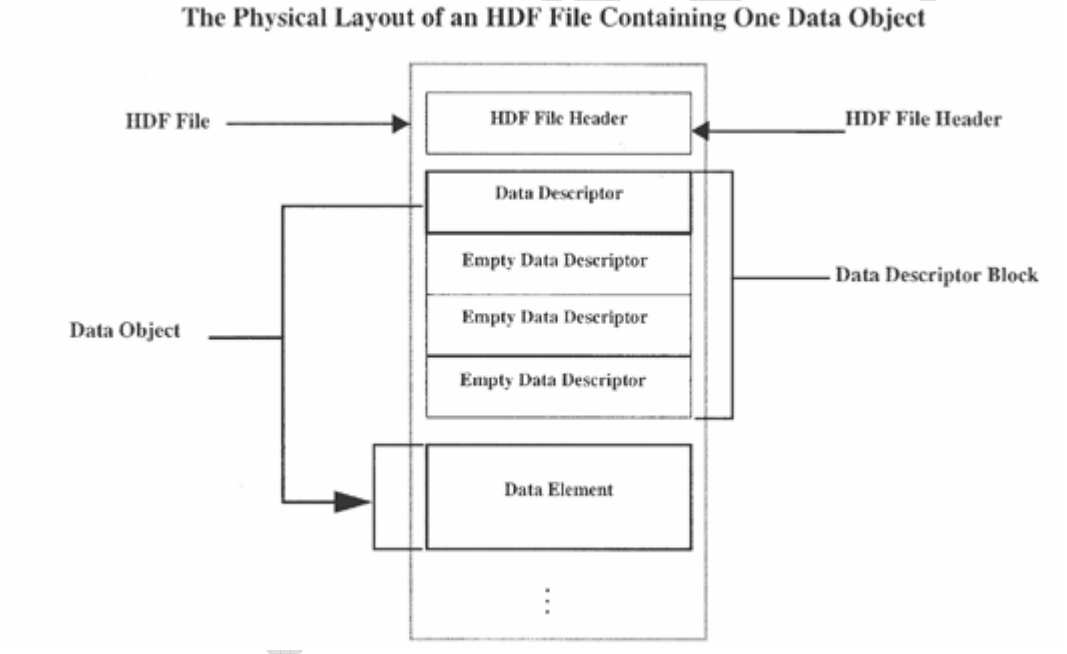


图 1d　包含一个数据对象的 HDF 文件结构

file id（文件号）是一个 32 比特的值，最多占用 HDF 文件的头 4 个字节。通过读取这个值，应用程序就知道此文件是否是一个 HDF 文件。

Data descriptor block（数据块描述符）包含一个数据描述符数值。所有的数据描述符都是 12 字节长，包含 4 个域，即一个 16 比特长的标签，一个 16 比特的引用字，一个 32 比特的数据偏移量和一个 32 比特的数据长度。如图 1e。

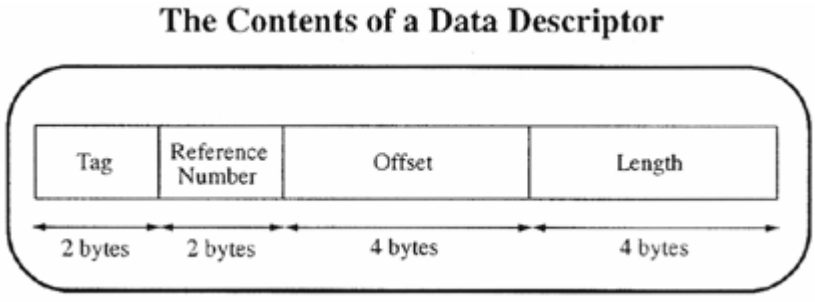


图 1e 数据描述符内容

tag（标记）是数据描述符域，表示存于相应数据内容的数据类型。例如 306 是光栅图像对象的识别符。

Reference number（引用号）是一个 16 比特无符号整型数。HDF 文件中的每一个对象，由 HDF 库和数据描述符中的标签确定一个唯一的引用字。在引用字确定的数据对象期间，标签和引用字不能改变。标签和引用字的结合可以唯一确定文件中对应的数据对象。

引用字没有必要连续指定，因此，在一个具有相同标签区分对象的方法后，不能假设引用字的值有任何意义。有时应用程序员也会发现在他们的程序中把一些另外的信息加到引用字中是很方便的，但必须强调的是，HDF 库本身并不识别这些含义。

Data offset field（数据偏移量）是一个 32 比特无符号整型字。通过存储文件开始时的字节数和数据内容开始时的字节数，指明文件中数据内容的位置。

Length field（长度域）是一个 32 比特无符号整型字。它表示整个数据内容的字节大小。数据内容增加，其长度也要增加。

Data element（数据成分）是数据对象的原始数据部分，包含每个象素的值。

1.7 HDF4 和 HDF5

值得注意的是，本文所介绍的信息都是关于 HDF4 的。新一代的 HDF5 是由 NCSA 于 1998 年发布。NCSA 支持 HDF4 并还将继续支持几年（细节如下）。HDF5 被设计为改善 HDF4 的一些局限性。HDF4 的某些局限性有：

- ✧ 单个文件不能存放多于 20000 个对象，单个文件大小也不能大于 2G 字节。
- ✧ 数据模式的兼容性不够好，有过多的对象类型，数据类型太严格。
- ✧ 库函数过时和过于复杂，不能有效地支持并行口的 I/O，很难用于线程应用中。

HDF5 包含如下的改进：

- ✧ 被设计为一种新的格式用来改进 HDF4.x，特别是每个文件可以存储更大的文件和更多的对象。
- ✧ 数据模式更简洁、更全面，它包含两个基本结构：多维数组记录结构，和分组结构。
- ✧ 更简洁、更利于工程库和应用编程接口，支持并行 I/O，线程和其他一些现代系统和应用要求。

虽然 HDF5 比 HDF4 有明显的优势，但 HDF4 仍然是在科学界使用得最为广泛。许多应用软件都是基于 HDF4 库开发的，许多数据产品是按 HDF4 格式生成的。把这些应用和数据产品从 HDF4 转换为 HDF5 需要花费时间和费用。因此，NCSA 仍然支持 HDF4 并还将持续几年。

第二章 HDF 库

2.1 本章简介

本章提供有关获取和安装 HDF 库的信息，介绍 HDF 应用编程接口和 HDF 支持的程序语言。

2.2 获取和安装 HDF 库

最新的 HDF4 版（HDF5 已经发布，但还没有广泛使用），HDF4.1r3 可以从 NCSA 的网站免费下载。这个版本的 HDF 库包括 C 和 FORTRAN 库文件、id 文件、HDF 命令行实用程序以及 HDF 文档。原代码和编译后的库都可以从 NCSA 那里得到。编译成的 HDF 库可用于 Windows 9x/NT、Macintonsh 和 Unix 平台。HDF 库支持的计算机平台和编译器有下列这些：

Operating platforms and the C Compilers and Fortran Compilers that run on those platforms

Platforms (OS)	C-Compiler	Fortran-Compiler
Sun4 (SunOS 4.1.4)	GCC 2.6.3	f77 SC1.0
Sun4 (Solaris 2.6)	CC	f77
SGI-Indy (IRIX v6.5)	CC 7.2.1	f77 7.2.1
SGI-Origin (IRIX64 v6.5-n32)	CC 7.2.1.2m	f77 7.2.1.2m
SGI-Origin (IRIX64 v6.5-64)	CC 7.2.1.2m	f77 7.2.1.2m
HP9000/755 (HP-UX B.10.20)	CC A.10.32.03	f77
Exemplar (HP-UX B.10.01)	CC V2.0	f77 V1.2.6
Cray T90 CFP (UNICOS 10.0.0eu u10.49)	CC 6.2.0.1	f90 3.2.0.1
Cray T90 IEEE (UNICOS 10.0.0ev d10.161)	CC 6.2.0.1	f90 3.2.0.1
Cray J90 (bob.0 10.0.0.3)	CC 6.2.0.1	f90 3.2.0.0
IBM SP (single node, v4.2)	XLC 3.1.4.10	f77 5.1.0.2
IBM AIX v4.3 (w/hdfi.h patch)	XLC 3.1.4.0	f77 3.2.5.0
DEC Alpha/Digital Unix v4.0	DEC C v5.6-079	Digital Fortran v5.1
DEC Alpha/Linux 2.0.36	cc	f77
DEC Alpha/OpenVMS AXP v6.2	DEC C v5.0-003	Digital Fortran 77 X7.1-156
DEC Alpha/OpenVMS AXP v7.1	DEC C v5.6-003	Digital Fortran 77 X7.1-156
VAX OpenVMS v6.2	DEC C v5.6	DEC Fortran v6.3
IBM PC - Intel Pentium (Solarisx86 2.5.1)	GCC 2.7.2	-
IBM PC - Intel Pentium (Linux(elf) 2.2.26)	GCC 2.8.1	g77 2.8.1
IBM PC - Intel Pentium (FreeBSD 3.1)	GCC 2.7.2.1	g77 2.7.2.1
PowerMac 7600/120 (MacOS 8.5.1)	MetroWerks CW1, CW4	-
Windows 9x/NT	MSVC++ 5.0	DEC Visual Fortran 5.0
DEC Alpha NT	MSVC++ 5.0	DEC Visual Fortran 5.0
T3E	CC 6.2.0.1	f90 3.2.0.0

表 2a 运行平台以及 C 和 FORTRAN 在这些平台上的编译器

为了把 HDF 软件包安装在你的计算机上，首先下载编译好了的 HDF 软件包，其次是解

包，然后再安装。头文件在 include 子目录里，lib 子目录里有 C 和 FORTRAN 库文件，bin 子目录有 HDF 命令行工具。特别需要注意是：使用 HDF 之前应将 bin 子目录路径加到环境变量里。HDF 命令行工具将在第 8 章进一步描述。安装和编译向导可以在 relase_notes 目录找到。

2.3 程序语言

HDF API 编程语言支持 C、FORTRAN 和 Java 程序语言。

HDF 库提供其接口例程的 C 和 FORTRAN-77 版本。所有的 HDF 库代码都是由 C 语言写成的。FORTRAN-77 到 C 转换程序能把所有 FORTRAN 参数数据类型转换为 C 数据类型，然后调用实施该接口例程功能的 C 例程。例如，FORTRAN 的 d8aimg 与 C 的 DFR8addimage 等价。执行相同的 C 代码时，调用这两个任意一个例程，都会把一个 8 比特的光栅图像加到 HDF 文件里。见图 2a。

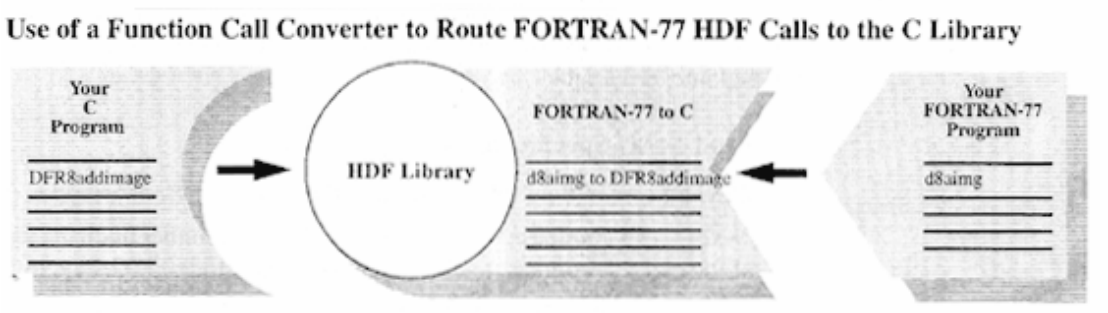


图 2a 使用 FORTRAN77 的 HDF 转换函数调用 C 库

用户在使用 HDF 文件时也可以用 Java 语言。Java 的 HDF 接口（JHI）提供一个与所有 HDF4.1r3 库函数连接的接口函数。JHI 是类似与 FORTRAN 的接口函数，已经作为 HDF 库的一部分而提供。通过使用 Java 包，任何 Java 的应用都能使用此接口类来读写 HDF 文件。这个 Java 包给标准的 HDF4.1r3 库包装了一个外壳，称为由 Java 到本体方法。Java 的任何应用程序都能使用 J HI。

2.4 应用编程接口（API）

HDF 库是应用编程接口的集合。这些应用编程接口是一组被应用程序调用的例程，可以对 HDF 文件里的 HDF 对象进行读、写和组织。第 1 章描述了一些为不同对象创建的特殊应用编程接口软件，它们是：

通用光栅图像 General Raster Image（GR API）存储、管理和获取光栅图像以及它们的维数和调色板。它也能灵活处理含于多个文件的独立调色板。要说明的是，操作 HDF 调色板对象的例程包含在 GR API 里。HDF 用户手册有更详细的介绍。

科学数据集 Scientific Data Set（SD API）在一个或一个以上的文件中，连同数据的字符或数字维数和属性，存储、管理和获取这些多维数组。

虚拟数据 Vdata（VS API）存储、管理和获取作为记录而存储于表格中的数据。

注解 Annotation（AN API）存储、管理和获取用来描述一个文件或包含该文件的任何数据对象的文本。这个接口同时对几个文件操作。

虚拟组合 Vgroup（V API）创建任何基本结构的 HDF 数据组。

2.5 头文件信息

除了在 SD 接口中调用例程的程序外，每个以 C 编写的 HDF 应用程序都必须包含头文件 hdf.h。相反，所有调用 SD 接口例程的程序都必须包含 id 文件 mfhdf.h。

使用 FORTRAN 编译器时必须将 hdf.inc 和 dffunc.inc 文件包含在内。假如 FORTRAN 编译器不支持这个包含文件，必须在 FORTRAN 程序中明确定义 HDF 库。HDF 库必须在 id 文件中。HDF 用户手册里有详细介绍。

2.6 编译指导

在编写和编译使用 HDF 库的程序之前，应阅读公布的有关文档。它随 HDF 库的编译信息一起分发，确保计算机平台有正确的编译器。如若还不能安装，则寻求系统管理员的帮助。

对于 UNIX 工作平台，按照如下方法编译程序，

C 语言：

```
cc –o <your program> <your program.c> –I<path for HDF include directory> \
   –L<path for HDF libraries> –lmfhdf –ldf –ljpeg –lz
or
cc –o <your program> <your program>.c –I<path for hdf include directory> \
   <path for libmfhdf.a> <path for libdf.a> \
   <path for libjpeg.a> <path for libz.a>
```

FORTRAN 语言：

```
f77 –o <your program> <your program>.f \
      –L<path for hdf libraries> –lmfhdf –ldf –ljpeg –lz
or
f77 –o <your program> <your program>.f \
       <path for libmfhdf.a> <path for libdf.a> \
       <path for libjpeg.a> <path for libz.a>
```

对于 WINDOWS 9x/NT：

Using Microsoft Visual C++ version 5.x or higher:

Under Tools–>Options, select the folder, Directories:

Under "Show directories for", select "Include files".

Add the following directories:

C:<path to HDF includes>\INCLUDE

Under "Show directories for", select "Library files":

Add the following directories:

C:<path to HDF libs>\LIB

Under Project–>Settings, select folder, Link:

Add the following libraries to the beginning of the list of

Object/Library Modules:

hd413.lib hm413.lib (single–threaded release version)

hd413d.lib hm413d.lib (single–threaded debug version)

hd413m.lib hm413m.lib (multi–threaded release version)

hd413md.lib hm413md.lib (multi–threaded debug version)

参见随 HDF 库一起发布的公布信息可以获得更详细的有关编译信息。

第三章 常规光栅图像（GR API）

3.1 本章简介

本章描述常规光栅（GR）数据模型，即 GR API。利用 GR API（应用编程接口）创建、写入、读取和查询光栅图像。

3.2 常规光栅图像数据模型

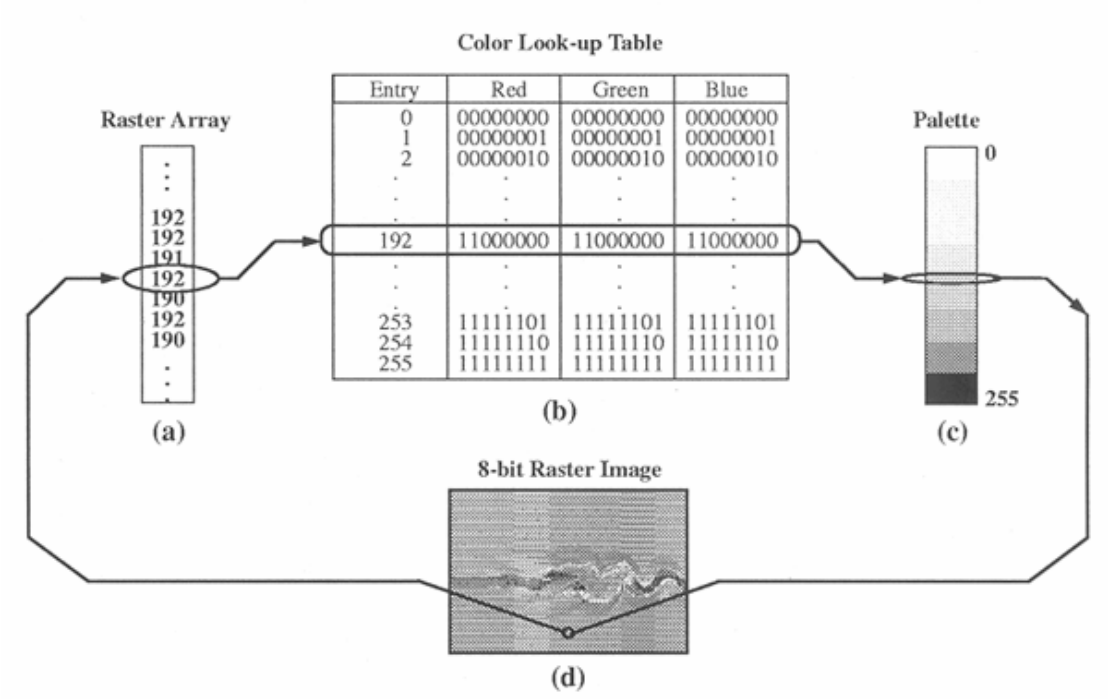


图 3a 光栅图像和调色板

常规光栅图像是一个数据结构，用于存储和描述光栅图像和与之关联的调色板。每一个 GR 对象必须包含如下内容：name（名字），dimensions（维数），image array（图像数组）和 pixel type（象素类型）。创建 GR 对象时，要求用户必须提供上述参数。

GR 对象中有两类可选内容：palettes（调色板）和 attributes（属性）。

GR 对象内的调色板也许是数据生成者的首选，用户会认为这能最有效地表达图像信息。

调色板的应用也用来增强那些不容易分辨的图像特征。没有附带调色板的 GR 图像，可通过 HDF 提供的配有标准调色板的可视化工具来显示。

数据对象属性是描述数据对象的特性或计划用途。例如，一个光栅图像对象的属性可用来定义缺省的象素值，当只有一部分图像数组有数据时，程序只把数据写入图像这部分，而其他部分则被写入缺省值。

参照表 3a 的各项描述。

Type	Component	Description
Required	Name	Name of the image object
Required	Dimensions	Width and height of the image
Required	Image Array	2D array of pixel values
Required	Pixel Type	HDF supported data types of pixel value
Optional	Palette	Color lookup tables attached to the image
Optional	Attribute	Auxiliary information about the raster image

表 3a 光栅图像成分

3.3 GR API

GR API 是由存储、获取和处理光栅图像对象的例程组成。GR API 里所有 C 的例程名字都有前缀“GR”，相对应 FORTRAN 的例程名都有前缀“mg”。访问任何光栅图像对象时，调用程序必须包含下列步骤或函数调用顺序：

Steps or Sequence of function calls to access the genereal raster image object.

Function Call Steps	C	FORTRAN
1. Open an HDF file.	Hopen	hopen
2. Initialize the GR interface.	GRstart	mgstart
3. Create or open a GR image.	GRcreate or GRselect	mgcreat or mgselct
4. Perform the desired operation.	GRwriteimage, GRreadimage, GRgetiminfo...	mgwrimg, mgrdimg, mggiinf...
5. Terminate access to the image.	GRendaccess	mgendac
6. Terminate access to the GR interface.	GRend	mgend
7. Close the HDF file.	Hclose	hclose

表 3b 常规光栅图像步骤或函数调用顺序

关于每个函数调用的详细信息见 HDF 第四版参考手册。

注意：除了使用 SD 接口外，其他使用 GR，VS，V 和 AN 接口的所有应用必须使用 Hopen 和 Hclose 例程来控制访问 HDF 文件。在一个应用程序中，HDF 文件是通过 Hopen 例程返回的文件标识符来进行访问的。接口程序只是用文件标识符来访问和处理文件。当对这个文件的所有操作完成时，结束这个应用前，必须通过调用 Hclose 程序来放弃这个文件标识符。

3.4 把光栅图像写入 HDF 文件

下列程序代码是把一个 10 x 8 的 8 比特图像写入一个被称为“example.hdf”的文件里。在此例中，用 GRcreate/mgcreat 来创建一个被称为“My image”的 GR 对象，用

Grwirteimage/mgwrimg 把下列象素值写入图像数组中。

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

C 语言：

```
--------------------------------------------------------------------------------
/* write_gr.c – write raster image to an HDF file */
#include "hdf.h"
#define WIDTH  10
#define HEIGHT8

main()
{
    intn   status;              /* status returned by function calls */
    int32 file_id,              /* HDF file identifier */
          gr_id,                /* GR interface identifier */
          ri_id;                /* raster image identifier */
    int32 dimensions[2] = {WIDTH, HEIGHT},   /* dimension size of the image array */
          start[2] = {0, 0};                 /* start position to write */
    /* Initialize the image array */
    uint8 image_array[HEIGHT][WIDTH] =     { 0,   1, 2,   3,   4, 5,   6, 7,   8,   9,
                                             10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
                                             20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
                                             30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
                                             40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
                                             50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
                                             60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
                                             70, 71, 72, 73, 74, 75, 76, 77, 78, 79};
    /* Create and open the file "example.hdf" */
    file_id = Hopen("example.hdf", DFACC_CREATE, 0);
    /* Initialize the GR interface */
    gr_id = GRstart(file_id);
    /* Create an raster image object */
    ri_id = GRcreate(gr_id, "My Image", 1, DFNT_UINT8, 0, dimensions);
    /* Write the image data */
    status = GRwriteimage(ri_id, start, NULL, dimensions, (VOIDP)image_array);
    /* Terminate access to the image object and the GR interface, close the HDF file */
    status = GRendaccess(ri_id);
    status = GRend(gr_id);
```

```
    status = Hclose(file_id);
}
```

Fortran 语言：

```
------------------------------------------------------------------------------
      program write_gr

      integer      WIDTH
      integer      HEIGHT
      integer      DFNT_UINT8
      integer      DFACC_CREATE
      parameter (WIDTH = 10,
     +     HEIGHT = 8,
     +     DFNT_UINT8 = 21,
     +     DFACC_CREATE = 4)
C     Function declaration
      integer      hopen, hclose
      integer      mgstart, mgcreat, mgwrimg, mgendac, mgend
C     Variable declaration
      integer      status
      integer      file_id
      integer      gr_id
      integer      ri_id
      integer      dimensions(2), start(2), stride(2)
      integer      image_array(HEIGHT, WIDTH)
      data  start  /0, 0/
      data  stride/1, 1/
      data  image_array      / 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
     +     10, 11, 12, 13, 14, 15, 16, 17, 18, 19
     +     20, 21, 22, 23, 24, 25, 26, 27, 28, 29
     +     30, 31, 32, 33, 34, 35, 36, 37, 38, 39
     +     40, 41, 42, 43, 44, 45, 46, 47, 48, 49
     +     50, 51, 52, 53, 54, 55, 56, 57, 58, 59
     +     60, 61, 62, 63, 64, 65, 66, 67, 68, 69
     +     70, 71, 72, 73, 74, 75, 76, 77, 78, 79/
C     Create and open the file
      file_id = hopen('example.hdf', DFACC_CREATE, 0)
C     Initialize the GR interface
      gr_id = mgstart(file_id)
C     Create an raster image object
      status = mgcreat(ri_id, 'My Image', 1, DFNT_UINT8, 0, dimensions)
C     Write the image data
      status = mgwrimg(ri_id, start, stride, dimensions, image_array)
C     Terminate access to the image and the GR interface, close the HDf file
      status = mgendac(ri_id)
```

```
status = mgend(gr_id)
status = hclose(file_id)
end
```

3.5 从 HDF 文件中读取光栅图像

下列程序代码是读取由上面例子创建的 8 比特图像的名字和维数，然后输出每个象素的值。Grgetiminfo/mggiinf 被用来获取有关 GR 对象的信息，用 GRreadimage/mgrdimg 例程把象素值读入缓冲区。

C 语言:

```
/* read_gr.c - read raster image from an HDF file */
#include "hdf.h"
#define WIDTH  10
#define HEIGHT8

main()
{
    intn   status;              /* status returned by function calls */
    int32 file_id,          /* HDF file identifier */
          gr_id,                /* GR interface identifier */
          ri_id,            /* rast image identifier */
          n_comps,          /* number of components an image contains */
          data_type,        /* data type of pixel values */
          interlace_mode,       /* image interlace mode */
          n_attrs,          /* number of image attributes */
          i, j;
    int32 dimensions[2],        /* dimensions of the image */
          start[2] = {0, 0}; /* start position to read from an image */
          uint8 image_array[HEIGHT][WIDTH]; /* pixel value array */
          char  name[MAX_GR_NAME];   /* image name */
    /* open the HDF file "example.hdf" for read */
    file_id = Hopen("example.hdf", DFACC_READ, 0);
    /* Initialize the GR interface */
    gr_id = GRstart(file_id);
    /* Get the image identifier of the first image */
    ri_id = GRselect(gr_id, 0);
    /* Get image information */
    status = GRgetiminfo (ri_id, name, &n_comps, &data_type,
                &interlace_mode, dimensions, &n_attrs);
    /* Read pixel values from the image */
    status = GRreadimage(ri_id, start, NULL, dimensions, (VOIDP)image_array);
    printf("Image Name: %s Image Size: %dx%d\n", name,
```

```
            dimensions[0], dimensions[1]);
        printf("Pixel Values:\n");
        /* Print out pixel values */
        for (i = 0; i < dimensions[1]; i++)
        {
        for (j=0; j < dimensions[0]; j++)
        printf("%5d", image_array[i][j]);
        printf("\n");
        }
        /* Terminate access to the image object and the GR interface, close the HDF file */
        status = GRendaccess (ri_id);
        status = GRend(gr_id);
        status = Hclose(file_id);
}
```

Fortran 语言：

```
--------------------------------------------------------------------------------
        program read_gr

        integer      WIDTH
        integer      HEIGHT
        integer      DFACC_READ
        parameter (WIDTH = 10, HEIGHT = 8, DFACC_READ = 1)
        integer      hopen, hclose
        integer      mgstart, mgselct, mggiinf, mgrdimg, mgendac, mgend
        integer      status
        integer      file_id
        integer      gr_id
        integer      ri_id
        integer      n_comps
        integer      data_type
        integer      interlace_mode
        integer      n_attrs
        integer      i, j
        integer      dimensions(2), start(2), stride(2)
        data  start  /0, 0/
        data  stride/1, 1/
        character*1image_array(HEIGHT, WIDTH)
        character*64      name
C       Open example.hdf for read
        file_id = hopen('example.hdf', DFACC_READ, 0)
C       Initialize the GR interface
        gr_id = mgstart(file_id)
C       Get the image identifier of the first image
        ri_id = mgselct(gr_id, 0)
```

```
C       Get image information
        status = mggiinf(ri_id, name, n_comps, data_type,
        interlace_mode, dimensions, n_attrs)
C       Read pixel values from from the image
        status = mgrdimg(ri_id, start, stride, dimensions, image_array)
        do 10 i = 1, dimensions[1]
        write(*, *) (image_array(i, j), j = 1, dimensions[2])
10      continue
C       Terminate access to the image and the GR interface, close HDF file
        status = mgendac(ri_id)
        status = mgend(gr_id)
        status = hclose(file_id)
        end
```

以下是这个程序的输出结果：

Image Name: My Image Image Size: 10 x 8

Pixel Values:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

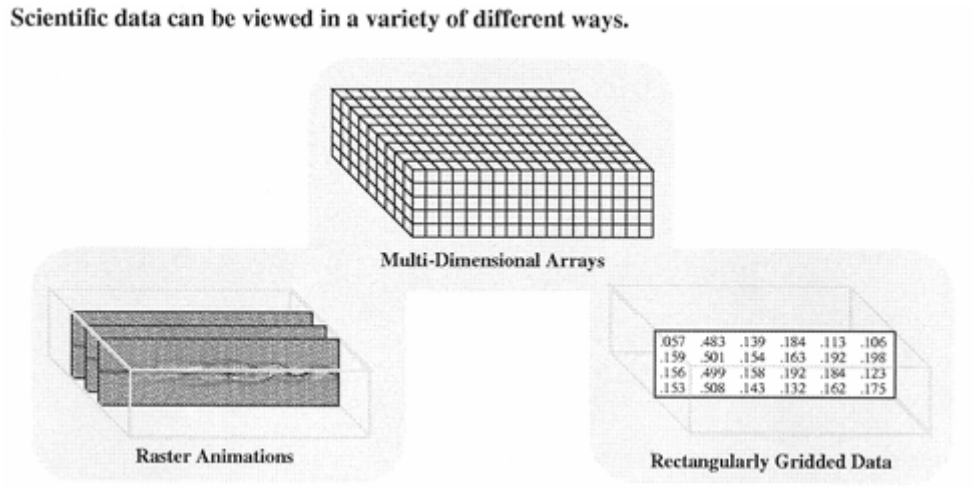
除上例使用的那些例程外，GR 接口提供了更多的例程。有关每个 GR 接口例程的更详细的描述，请参见 HDF 用户指南和 HDF 参考手册。

第四章 科学数据集（SD API）

4.1 本章简介

本章描述科学数据集（SD）数据模型，即 SD API。本章介绍用来创建、写入、读取和从科学数据集查询信息的 SD 接口例程。

4.2 科学数据集数据模型



科学数据集（SDS）是一个用来存储和描述科学数据多维数组的数据结构。每个 SDS 必须包含一个 SDS 数组，一个名字，一个数据类型和 SDS 数组的维数。SDS 可选组件有三类：预定义属性，用户定义属性和维数尺度。仅当调用程序且特别声明时才会创建这些可选组件。SDS 的内容见表 4a。

SDS required and optional components

Type	Component	Description
Required	Name	Name of a SDS object
	Dimensions	Specify the shape and size of an SDS array
	Data Type	Data type of data elements of SDS array
	SDS Array	Multidimensional array of scientific data
Optional	Predefined Attributes	Reserved Attributes defined by HDF library
	User–defined Attributes	Auxiliary information about SDS array or dimension defined by the user.
	Dimension Scales	Sequence of numbers placed along a dimension to demarcate intervals along it

图 4a 科学数据集的必选和可选项

4.3 SD API

SD 接口提供存储、读取和处理 SDS 的例程。SD 接口的所有 C 例程都是以“SD”前缀开头的。对应 FORTRAN 的例程以“sf”开头。访问 SDS 对象时，调用程序必须包含下列步骤或函数调用顺序。

Steps or Sequence of function calls to access the SDS object.

Function Call Steps	C	Fortran
1. Open a file and initialize the SD interface	SDstart	sfstart
2. Create or open a SDS object	SDcreate or SDselect	sfcreate or sfselect
3. Perform desired operation	SDwritedata, SDreaddata, SDgetinfo, SDfileinfo ...	sfwdata, sfrdata, sfginfo, sffinfo ...
4. Terminate access to the SDS	SDendaccess	sfendacc
5. Terminate access to the SD interface and close the file	SDend	sfend

图 4b　访问科学数据集对象的步骤或函数调用顺序

注意：在 SD 接口中，用 SDstart 打开 HDF 文件，而不用 Hopen。用 SDend 来关闭 HDF 文件。使用 SD 接口例程的 C 程序必须含有头文件“mfhdf.h”。

4.4 把科学数据集写入 HDF 文件

下列程序代码在“example.hdf”文件中创建一个称为“My SDS”的 SDS 对象。在此例中，用 SDcreate/sfcreate 创建该 SDS 对象，用 SDwritedata/sfwdata 把一个有 64 位浮点数的 2x3x5 数组写入该 SDS 对象。这个三维数组由下列数值初始化。

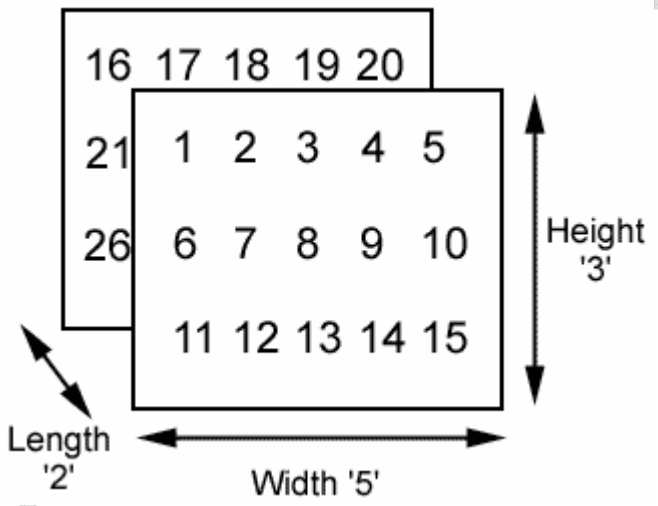


图 4c　2 x 3 x 5 数组

C:

```
/* write_sds.c - write scientific data set to an HDF file */

#include "mfhdf.h"
#define WIDTH        5
#define HEIGHT       3
#define LENGTH       2
#define RANK 3

main()
```

```
{
    intn     status;  /* status returned by function calls */
    int32 sd_id,      /* SD interface identifier */
    sds_id;           /* SDS object identifier */
    int32 dimensions[3] = {LENGTH, WIDTH, HEIGHT},
    start[3] = {0, 0, 0},
    stride[3] = {1, 1, 1};
    /* Initialize the sds array */
    float64 sds_array[LENGTH][HEIGHT][WIDTH] =
    { 1.0, 2.0, 3.0, 4.0, 5.0,
    6.0, 7.0, 8.0, 9.0, 10.0,
    11.0, 12.0, 13.0, 14.0, 15.0,
    16.0, 17.0, 18.0, 19.0, 20.0,
    21.0, 22.0, 23.0, 24.0, 25.0,
    26.0, 27.0, 28.0, 29.0, 30.0 };
    /* Create the file and initialize the SD interface */
    sd_id = SDstart("example.hdf", DFACC_CREATE);
    /* Create the SDS object */
    sds_id = SDcreate(sd_id, "my SDS", DFNT_FLOAT64, RANK, dimensions);
    /* Write the data to SDS array */
    status = SDwritedata(sds_id, start, stride, dimensions,
(VOIDP)sds_array);
    /* Terminate access to the SDS object and the SD interface, close the
file */
    status = SDendaccess(sds_id);
    status = SDend(sd_id);
}
```

Fortran:

```
program write_sds

    integer WIDTH
    integer HEIGHT
    integer LENGTH
    integer RANK
    integer DFACC_CREATE
    integer DFNT_FLOAT64
    parameter ( WIDTH = 5,
    +        HEIGHT = 3,
    +        LENGTH = 2,
    +        RANK = 3,
    +        DFACC_CREATE = 4,
    +        DFNT_FLOAT64 = 6)
C   Function declaration
```

```
      integer sfstart, sfselect, sfwdata, sfendacc, sfend
C     Variable declaration
      integer status, sd_id, sds_id
      integer dimensions(3), start(3), stride(3)
      double precision sds_array(LENGTH, HEIGHT, WIDTH)
      data dimensions /LENGTH, WIDTH, HEIGHT/
      data start /0, 0, 0/
      data stride /1, 1, 1/
      data sds_array /1.0, 2.0, 3.0, 4.0, 5.0,
     +        6.0, 7.0, 8.0, 9.0, 10.0,
     +        11.0, 12.0, 13.0, 14.0, 15.0,
     +        16.0, 17.0, 18.0, 19.0, 20.0,
     +        21.0, 22.0, 23.0, 24.0, 25.0,
     +        26.0, 27.0, 28.0, 29.0, 30.0/
C     Create the file and initialize the SD interface
      sd_id = sfstart('example.hdf', DFACC_CREATE)
C     Create the SDS object
      sds_id = sfcreate(sd_id, 'My SDS', DFNT_FLOAT64, RANK, dimensions)
C     Write the data to SDS array
      status = sfwdata(sds_id, start, stride, dimensions, sds_array)
C     Terminate access to the SDS object and the SD interface, close the file
      status = sfendacc(sds_id)
      status = sfend(sd_id)
```

4.5 从 HDF 文件读取科学数据集

下列程序例子是从名为“My SDS”的 SDS 对象中读取信息，“My SDS”是由上述例子“example.hdf”文件创建的。在此例中，用 SDgetinfo/sfginfo 获取有关这个 SDS 对象的数据类型和结构信息，用 SDreaddata/sfrdata 从 SDS 数组读取数据值。

C:

```
/* read_sds.c - read scientific data set from an HDF file */
#include "mfhdf.h"
#define LENGTH 2
#define HEIGHT 3
#define WIDTH 5

main()
{
    intn status;      /* status returned by function calls */
    int32 sd_id,      /* SD interface identifier */
    sds_id,           /* SDS object identifier */
    n_datasets,       /* number of SDS in the file */
```

```
    n_file_attrs,    /* number of file attributes */
    rank,            /* number of dimensions of the SDS */
    data_type,       /* type of data element */
    n_attrs, /* number of attributes */
    i, j, k;
    int32 dimensions[3],
    start[3] = {0, 0, 0},     /* start position to read from SDS */
    stride[3] = {1, 1, 1};    /* stride to move forward for next read */
    char name[MAX_NC_NAME];   /* SDS object name */
    float64 sds_array[LENGTH][HEIGHT][WIDTH]; /* array to hold the data
elements */
    /* Open the HDF file "sds.hdf" for read */
    sd_id = SDstart ("example.hdf", DFACC_READ);
    /* How many SDS objects and file attributes are in this file? */
    status = SDfileinfo(sd_id, &n_datasets, &n_file_attrs);
    printf("number of datasets = %d\n", n_datasets);
    printf("number of file attributes = %d\n", n_file_attrs);
    /* Select the first SDS object */
    sds_id = SDselect(sd_id, 0);
    /* Get information about this SDS object */
    status = SDgetinfo(sds_id, name, &rank, dimensions,
    &data_type, &n_attrs);
    printf("name = %s\n", name);
    printf("rank = %d\n", rank);
    printf("dimensions = ");
    for (i = 0; i < rank; i++) printf("%d ", dimensions[i]);
    printf("\nData elements are: \n");
    /* Read data from the SDS object */
    status = SDreaddata(sds_id, start, stride, dimensions,
(VOIDP)sds_array);
    /* Print out the data elements */
    for (i = 0; i < dimensions[0]; i++) {
            for (j = 0; j < dimensions[1]; j++) {
                    for (k = 0; k < dimensions[2]; k++)
                    printf(" %f", sds_array[i][j][k]);
                    printf("\n");
                    }
            printf("\n");
    }
    /* Terminate access to the SDS object and the SD interface, close the
file */
    status = SDendaccess(sds_id);
    status = SDend(sd_id);
}
```

Fortran:

```
program read_sds
    integer LENGTH
    integer HEIGHT
    integer WIDTH
    integer DFACC_READ

    parameter (LENGTH = 2,
    +        HEIGHT = 3,
    +        WIDTH = 5,
    +        MAX_NC_NAME = 1)
C   Function declaration
    integer sfstart, sffinfo, sfginfo, sfselect
    integer sfrdata, sfendacc, sfend

C   Vaiable declaration
    integer status, sd_id, sds_id
    integer n_datasets, n_file_attrs, rank, data_type, n_attrs
    integer i, j, k
    integer dimensions(3), start(3), stride(3)
    character*256 name
    double presision sds_array(LENGTH, HEIGHT, WIDTH)
    data start /0, 0, 0/, stride /1, 1, 1/
C   Open the HDF file "sds.hdf" for read
    sd_id = sfstart('example.hdf', DFACC_READ)
C   How many SDS objects and file attributes are in this file
    status = sffinfo(sd_id, n_datasets, n_file_attrs)
C   Select the first SDS object
    sds_id = sfselect(sd_id, 0)
C   Get information about this SDS object
    status = sfginfo(sds_id, name, rank, dimensions, data_type, n_attrs)
    write(*, *) "name = ", name(1:15)
    write(*, *) "rank = ", rank
    write(*, *) "dimensions = ", (dimensions(i), i = 1, rank)
    write(*, *) "Data elements are: "
C   Read data from the SDS object
    status = sfrdata(sds_id, start, stride, dimensions, sds_array)
C   Print out the data elements
    do 20 i = 1, dimensions(1)
    do 10 j = 1, dimensions(2)
    write(*,*) (sds_array(i, j, k), k = 1, dimensions(3)
    10 continue
    20 continue
```

```
C   Terminate access to the SDS object
    status = sfendacc(sds_id)
C   Terminate the SD interface, close the file
    status = sfend(sd_id)
    end
```

如下是此程序的输出结果：

```
number of datasets = 1
number of file attributes = 0
name = My SDS
rank = 3
dimensions = 2  3  5
Data elements are:
1.000000  2.000000  3.000000  4.000000  5.000000
6.000000  7.000000  8.000000  9.000000  10.000000
11.000000 12.000000 13.000000 14.000000 15.000000
16.000000 17.000000 18.000000 19.000000 20.000000
21.000000 22.000000 23.000000 24.000000 25.000000
26.000000 27.000000 28.000000 29.000000 30.000000
```

注意：除上述使用的那些例程外，SD 接口提供了更多的例程。有关每个 SD 接口例程更为详细的描述，请参见 HDF 用户指南和 HDF 参考手册。

第五章 Vdata（VS API）

5.1 本章简介

本章描述 Vdata 数据模型和 VS API，并介绍用于创建、写入、读取和查询 Vdata 的 Vdata 接口例程。

5.2 Vdata 数据模型

HDF Vdata 模型提供用于在 HDF 文件中存储定制的表格或 Vdata。一个 Vdata 象一个表格，其记录值存放于这个固定记录长度域的表格中。每一个域有其自己的大小和数据类型。所有的记录数有相同的结构，每个域中所有的值有相同的数据类型。Vdata 由一个名字，一个类和一组单独的域名确定。（见图 5a）

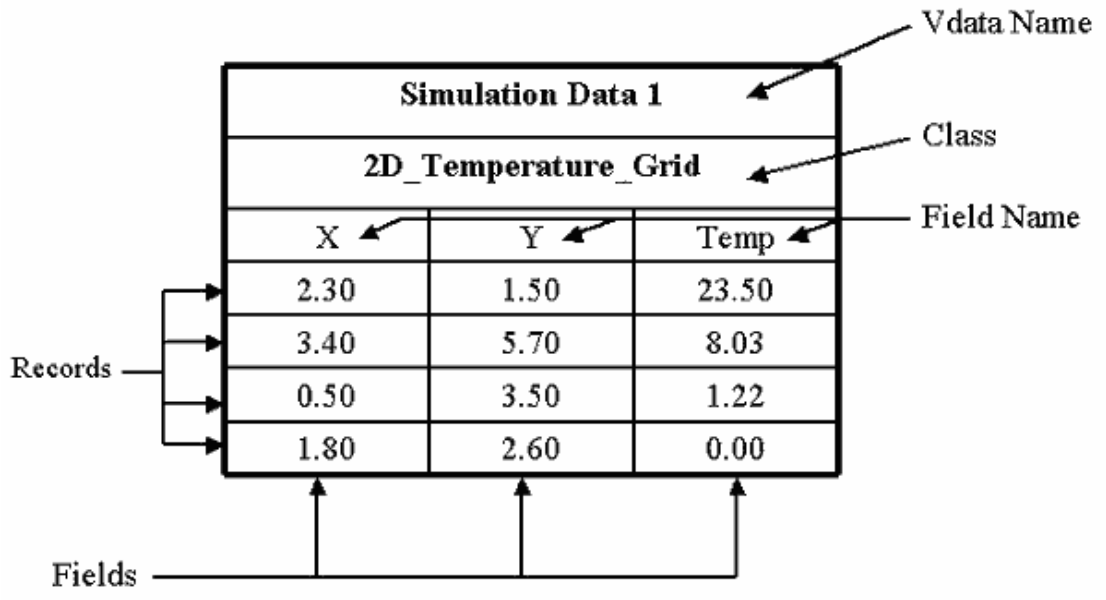


图 5a　Vdata 表格结构

一个 Vdata 名是一个描述表格原点或内容的标号。在含有多个表格文件中，它常作为搜索名。通过识别数据的内涵，Vdata 类进一步区别个别的表格。类常常被用来确定怎样解释一个记录的域。最后，Vdata 识别（域）名组成一个记录的独立域。

5.3 VS API

VS 接口提供定义、组织和处理 Vdata 的例程。C 程序中，Vdata 的例程是以“VS”，“VF”，“VSQ”和“VH”前缀开始的。对应的 FORTRAN 例程则以“vsf”，“vf”，“vsq”和“vh”开始。VS/vsf 例程执行最通用的 vdata 操作，VF/vf 例程查询关于 vdata 域的信息，VSQ/vsq 例程则查询有关特定的 vdatas 信息。VH/vh 例程是高级程序，写入单一域的 vdatas。

为了访问 Vdata，调用程序必须包含下列步骤或函数调用顺序：

Steps or Sequence of function calls to access a Vdata.

Function Call Steps	C	Fortran
1. Open the HDF file	Hopen	hopen
2. Initialize the Vdata interface	Vstart	vfstart
3. Create or open a Vdata object	VSattach	vsfatch
4. Perform the desired operations	VSwrite, VSread, VSsetname, VSsetclass, VSfdefine, VSsetfields, VSinquire, VSsetname, VSsetclass, VSfdefine, VSsetfields, VSinquire ...	vsfwrt, vsfrd, vsfsnam, vsfscls, vsffdef, vsfsfld, vsfinq, vsfsnam, vsfscls, vsffdef, vsfsfld, vsfinq ...
5. Terminate access to the Vdata object	VSdetach	vsfdtch
6. Terminate access to the Vdata interface	Vend	vfend
7. Close the file	Hclose	hclose

图 5a　Vdata 步骤或函数调用顺序

每个函数调用的更详细介绍见 HDF4 参考手册。

5.4 把 Vdata 写入 HDF 文件

下列程序中，名为“Simulation Data 1”的 Vdata 被创建来存储文件“example.hdf”中的表格 5a 表示的数据记录。VS 例程，如用 VSattach/vsfatch 创建这个 vdata，用 VSsetname/vsfsnam 和 VSsetclass/vsfscls 来指定名字和类，用 VSfdefine/vsffdef 定义每个域的数据类型和大小。用 VSwrite/vsfwrt 把数据记录写入这个 vdata。

C:

```
/* write_Vdata.c - write a Vdata to an HDF file */
#include "hdf.h"

main()
{
    intn     status;          /* statuts returned by function calls */
    int32    file_id, /* HDF file identifier */
             Vdata_id,        /* Vdata identifier */
             n_records;       /* number of records in the Vdata */
    /* Initialize the values to be write into the Vdata table */
    float32 data_buf[4][3] = { 2.30, 1.50, 23.50,
    3.40, 5.70, 8.03,
    0.50, 3.50, 1.22,
    1.80, 2.60, 0.00);
    /* Open hdf file "example.hdf" for writing */
    file_id = Hopen("example.hdf", DFACC_WRITE, 0};
    /* Initialize the VS interface */
    status = Vstart(file_id);
    /* Create a new Vdata object */
    Vdata_id = VSattach(file_id, -1, "w");
    /* Give the name and class of this Vdata */
    status = VSsetname(Vdata_id, "Simulation Data 1");
    status = VSsetclass(Vdata_id, "2D_Temperature_Grid");
    /* Define the name, data_type and order of each field */
    status = VSfdefine(Vdata_id, "X", DFNT_FLOAT32, 1);
    status = VSfdefine(Vdata_id, "Y", DFNT_FLOAT32, 1);
    status = VSfdefine(Vdata_id, "Temp", DFNT_FLOAT32, 1);
    /* Specify the Vdata fields to be written */
    status = VSsetfields(Vdata_id, "X,Y,Temp");
    /* Write records into the vata */
    n_records = VSwrite(Vdata_id, (uint8 *)data_buf, 3, FULL_INTERLACE);
    /* Terminate the access to this Vdata */
    status = VSdetach(Vdata_id);
    /* Terminate the access to the VS interface and close the file */
```

```
    status = Vend(file_id);
    status = Hclose(file_id);
}
```

Fortran:

```
      program write_Vdata.c

      integer DFACC_WRITE
      integer DFNT_FLOAT32
      parameter ( DFACC_WRITE = 2,
     +        DFNT_FLOAT32 = 5,
     +        FULL_INTERLACE = 0)
C     Fuction declaration
      integer hopen, hclose
      integer vfstart, vsfatch, vsfsnam, vsfscls, vsffdef
      integer vsfsfld, vsfwrt, vfend
C     Variable declaration
      integer status, file_id, Vdata_id, n_records
      real data_buf
C     Initialize the values to be write
      data data_buf /2.30, 1.50, 23.50,
     +        3.40, 5.70, 8.03,
     +        0.50, 3.50, 1.22,
     + 1.80, 2.60, 0.00/
C     Open hdf file "example.hdf" for writing
      file_id = hopen('example.hdf', DFACC_WRITE, 0);
C     Initialize the VS interface
      status = vfstart(file_id)
C     Create a new Vdata object
      Vdata_id = vsfatch(file_id, -1 , 'w')
C     Give the name and class of this Vdata
      status = vsfsnam(Vdata_id, 'Simulation Data 1')
      status = vsfscls(Vdata_id, '2D_Temperature_Grid')
C     Dfine the name, data_type and order of each field
      status = vsffdef(Vdata_id, 'X', DFNT_FLOAT32, 1)
      status = vsffdef(Vdata_id, 'Y', DFNT_FLOAT32, 1)
      status = vsffdef(Vdata_id, 'Temp', DFNT_FLOAT32, 1)
C     Specify the Vdata fields to be written
      status = vsfsfld(Vdata_id, 'Longitude,Latitude,Temperature')
C     Write records into the Vdata
      n_records = vsfwrt(Vdata_id, data_buf, 4, FULL_INTERLACE)
C     Terminate the access th this Vdata
      status = vsfdtch(Vdata_id)
C     Terminate the access to the VS interface and close the file
```

```
status = vfend(file_id)
status = hclose(file_id)
end
```

5.5 从 HDF 文件中读取 Vdata

如下的程序代码例子中，用 VSinquire（vsfinq）获取一个 Vdata 的表格名字、大小、域名列表和记录数的信息。用 VSread（vsfrd）把数据记录从 Vdata 对象中读出，然后读到数据缓冲区。

C:

```
/* read_Vdata.c - obtain information from a Vdata */
#include "hdf.h"

main()
{
    intn status;      /* status returned by function calls */
    int32 file_id,    /* HDF file identifier */
    Vdata_ref,        /* Vdata index number */
    Vdata_id,         /* Vdata identifier */
    n_records,        /* number of records in a Vdata */
    interlace_mode,   /* interlace_mode of records stored in a Vdata*/
    Vdata_size,       /* size of a Vdata record */
    i, j;
    float32 data_buf[4][3];   /* data buffer to read in the Vdata records
*/
    char fieldname_list[50], /* names of Vdata fields separated by "," */
    Vdata_name[VSNAMELENMAX];
    /* Open the HDF file "example.hdf" for reading */
    file_id = Hopen("example.hdf", DFACC_READ, 0);
    /* Initialize the VS interface */
    status = Vstart(file_id);
    /* Find the reference number of the first Vdata */
    Vdata_ref = VSgetid(file_id, -1);
    /* Attach to the Vdata for reading */
    Vdata_id = VSattach (file_id , Vdata_ref, "r");
    /* Obtain Information about this Vdata */
    status = VSinquire(Vdata_id, &n_records, &interlace_mode,
    fieldname_list, &Vdata_size, Vdata_name);
    /* Print Vdata information */
    printf("Vdata_name = %s\n", Vdata_name);
    printf("Vdata_size = %d bytes\n", Vdata_size);
    printf("fieldname_list = %s\n", fieldname_list);
```

```
        printf("number_of_records = %d\n", n_records);
        printf("interlace_mode = %s\n", interlace_mode ? "NONE" : "FULL");
        printf("records: \n");
        /* Specify the fields to be read */
        status = VSsetfields(Vdata_id, fieldname_list);
        /* Read the Vdata records into buffer */
        n_records = VSread(Vdata_id, (uint8*) data_buf, n_records,
        interlace_mode);
        /* Print Vdata records */
        for (i = 0; i < 4; i++)
        {
                for (j = 0; j < 3; j++)
                printf("%f ", data_buf[i][j]);
                printf("\n");
        }
        /* Terminate the access to the Vdata */
        status = VSdetach(Vdata_id);
        /* Terminate the access to the VS interface and clost the file */
        status = Vend(file_id);
        status = Hclose(file_id);
}
```

Fortran:

```
program read_Vdata
    integer DFACC_READ
    integer VSNAMELENMAX
    parameter (DFACC_READ = 1)
C   Function declaration
    integer hopen, hclose
    integer vfstart, vsffnd, vsfatch, vsfinq
    integer vsfsfld, vsfrd, vsfdtch, vfend
C   Variable Declaration
    integer status
    integer file_id, Vdata_ref, Vdata_id
    integer n_records, interlace_mode, Vdata_size
    integer i, j
    real    data_buf(4)(3)
    character*50 fieldname_list
    character*64 Vdata_name
C   Open the HDF file "example.hdf" for reading
    file_id = hopen('example.hdf', DFACC_READ, 0)
C   Initialize the VS interface
    status = vfstart(file_id)
```

```
C   Get the reference number of the first Vdata
    Vdata_ref = vsfgid(file_id, -1);
C   Attach to the Vdata for reading
    Vdata_id = vsfatch(file_id, Vdata_ref, 'r')
C   Obtain Information about this Vdata
    status = vsfinq(Vdata_id, n_records, interlace_mode,
    fieldname_list, Vdata_size, Vdata_name)
C   Print Vdata information
    write(*,*) 'Vdata_name = ', Vdata_name
    write(*,*) 'vadata_size = ', Vdata_size
    write(*,*) 'fieldname_lsit = ', fieldname_list
    write(*,*) 'number_of_records = ', n_records
    if (interlace_mode .eq. 0) then
            write(*,*) 'interlace_mode = FULL'
    else
            write(*,*) 'interlace_mode = NONE'
            write(*,*) 'records:'
C   Specify the fields to be read
    status = vsfsfld(Vdata_id, fieldname_list)
C   Read the Vdata records into buffer
    n_records = vsfrd(Vdata_id, data_buf, n_records, interlace_mode)
C   Print Vdata records
    do 10 i = 1, 3
    write(*,100) (data_buf(i, j), j = 1, 3)
10  continue
1000format(3F9.6)
C   Terminate the access to the Vdata
    status = vsfdtch(Vdata_id)
C   Terminate the access to the VS interface and close the file
    status = vfend(file_id)
    status = hclose(file_id)
    end
```

如下是此例程的计算输出结果：

```
Vdata_name = Simulation Data 1
Vdata_size = 12 bytes
fieldname_list = X,Y,Temp
number_of_records = 4
interlace_mode = FULL
records:
2.300000 1.500000 23.500000
3.400000 5.700000 8.030000
0.500000 3.500000 1.220000
1.800000 2.600000 0.000000
```

注意：除了上述例子中使用的那些 VS 接口例程，VS 接口还提供更多的例程，有关每个 VS 接口例程的详细介绍请参照 HDF 用户手册和 HDF 参考手册。

第六章 注解接口（AN API）

6.1 本章简介

本章介绍 HDF 的注解接口（AN），AN API。将介绍用于存储和获取 HDF 文件和对象注解的注解接口例程。

6.2 注解数据模式

注解是元数据，用于描述一个 HDF 文件或它包含的任何数据要素。注解是用于解释文件或数据对象的文本字符串。注解可以短到一个名字，或长到一段程序代码。例如，假如数据来源于卫星，注解就可能包括数据源，相关的环境条件或其它一些相关信息。在一个假想的黑洞情况下，注解可能包括生成这个数据的程序源代码。

注解主要分为两种大类：标签和描述。标签是一种短形式的注解，主要用于把诸如指定标题或时间印记到文件或其它数据对象中。长的注解被称为描述，通常包含更为广泛的信息，如源代码模块或数学公式。有四种注解形式：文件标签，文件描述，对象标签和对象描述，见图 6a。

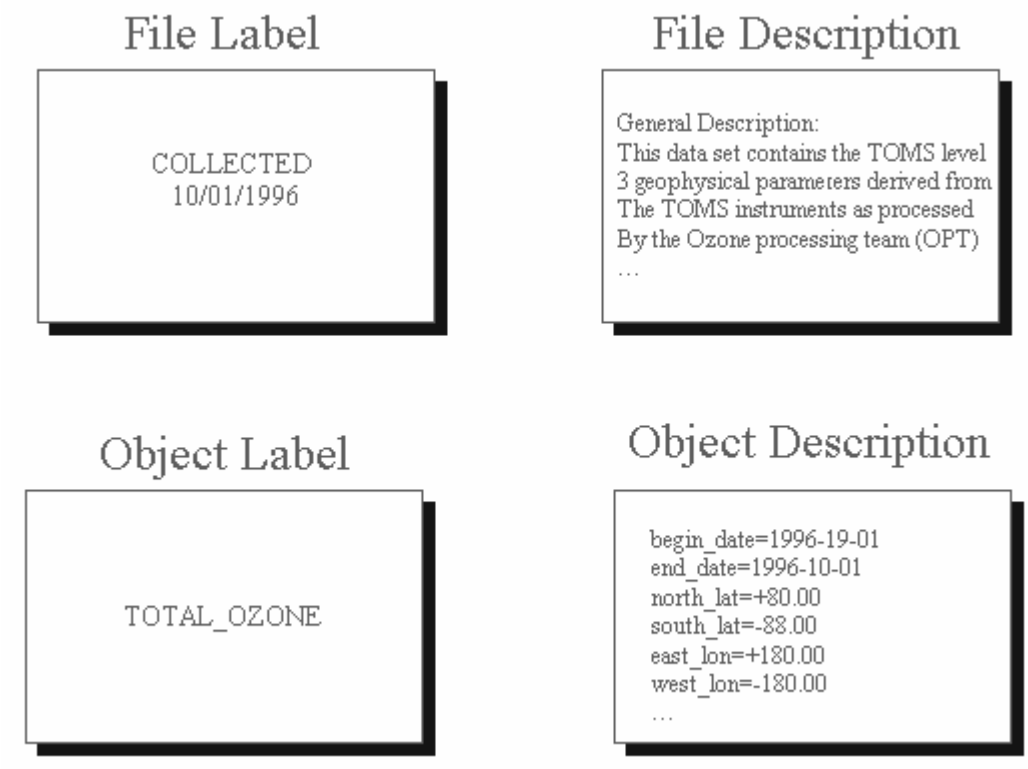


图 6a　HDF 注解类型

6.3 AN API

AN API 提供一些例程，用于把注解存于 HDF 文件或从 HDF 文件获取注解。AN 接口的所有 C 例程都有前缀“AN”，对应的 FORTRAN 例程名的前缀为“af”。调用程序必须包含如下步骤或函数调用顺序。

Steps or Sequence of function calls to access an annotation.

Function Call Steps	C	Fortran
1. Open the HDF file	Hopen	hopen
2. Initialize the AN interface	ANstart	afstart
3. Create or open an annotation	ANcreate or ANselect	afcreate or afselect
4. Perform the desired operations	ANwriteann, ANreadann ...	afwriteann, afreadann ...
5. Terminate access to the annotation	ANendaccess	afendaccess
6. Terminate access to the AN interface	ANend	afend
7. Close the HDF file	Hclose	hclose

表 6a　注解步骤或函数调用

每个详细的函数调用信息见 HDF4 参考手册。

6.4 把注解写入 HDF 文件

下列程序代码是把一个文件标签和一个文件描述注解写入文件“example.hdf”，此文件是有前面章节的代码例子生成的。在此例中，ANcreatef/affcreate 用来创建文件注解，ANwriteann/afwriteann 用来写这些注解。

C:

```
/* write_an.c - write annotations to an HDF file */
#include "hdf.h"
#define FILE_LABEL_TXT  "This is a file label"
#define FILE_DESC_TXT   "This is a file description"

main()
{
    int32 status,    /* status returned by function calls */
          file_id,   /* HDF file identifier */
          an_id,     /* AN interface identifier */
          label_id,  /* file label identifier */
          desc_id;   /* file description identifier */
    /* Open "example.hdf" for writing */
    file_id = Hopen ("example.hdf", DFACC_WRITE, 0);
```

```
    /* Initialize the AN interface */
    an_id = ANstart(file_id)
    /* Create a file label annotation */
    label_id = ANcreatef(an_id, AN_FILE_LABEL);
    /* Write the file label annotation */
    status = ANwriteann(label_id, FILE_LABEL_TXT, strlen(FILE_LABEL_TXT));
    /* Create a file description annotation */
    desc_id = ANcreatef(an_id, AN_FILE_DESC);
    /* Write the file description annotation */
    status = ANwriteann(desc_id, FILE_DESC_TXT, strlen(FILE_DESC_TXT));
    /* Terminate access to annotations */
    status = ANendaccess(label_id);
    status = ANendaccess(desc_id);
    /* Terminate access to the AN interface and close the HDF file */
    status = ANend(an_id);
    status = Hclose(file_id);
}
```

Fortran:

```
      program write_an
      character*20 FILE_LABEL_TXT
      character*26 FILE_DESC_TXT
      parameter (FILE_LABEL_TXT = "This is a file label",
     +           FILE_DESC_TXT  = "This is a file description")
      integer DFACC_WRITE
      integer AN_FILE_LABEL
      integer AN_FLIE_DESC
      parameter (DFACC_WRITE = 2
     +         AN_FILE_LABEL = 2,
     +         AN_FILE_DESC = 3)
C     Function declaration
      integer hopen, hclose
      integer afstart, affcreate, afwriteann
      integer afendaccess, afend
C     Variable declaration
      integer status, file_id, an_id
      integer label_id, descr_id
C     Open file "example.hdf" for writing
      file_id = hopen('example.hdf', DFACC_WRITE, 0)
C     Initialize the AN interface
      an_id = afstart(file_id)
C     Create a file label annotation
      label_id = affcreate(an_id, AN_FILE_LABEL)
```

```
C   Write the file label annotation
    status = afwriteann(label_id, FILE_LABEL_TXT, len(FILE_LABEL_TXT))
C   Create a file description annotation
    desc_id = affcreate(an_id, AN_FILE_DESC)
C   Write the file description annotation
    status = afwriteann(desc_id, FILE_DESC_TXT, len(FILE_DESC_TXT))
C   Terminate access to annotations
    status = afendaccess(label_id)
    status = afendaccess(desc_id)
C   Terminate access to the AN interface and close the file
    status = afend(an_id)
    status = hclose(file_id)
    end
```

6.5 从 HDF 文件读取注解

下列例子从一个 HDF 文件获得注解信息，并读取这些注解的内容。在此例中，用 ANfileinfo/affileinfo 获取存于该 HDF 文件中四种注解的每一个注解编号。用 ANannlen/afanlen 获得注解的大小，用 ANreadann/afreadann 把注解读入到字符缓冲区。

C:

```
/* read_an.c - read annotations from an HDF file */
#include "hdf.h"

main()
{
    int32 status,    /* status returned by function calls */
    file_id, /* HDF file identifier */
    an_id,   /* AN interface identifier */
    n_file_labels,   /* number of file labels */
    n_file_descs,    /* number of file descriptions */
    n_object_labels, /* number of object descriptions */
    n_object_descs,  /* number of object descriptions */
    file_label_id,   /* file label annotaion identifier */
    file_desc_id,    /* file description annotation identifier*/
    file_label_len,  /* length of file label annotation */
    file_desc_len;   /* length of file description annotation */
    char *file_label_buf, /* buffer to hold file label text */
    *file_desc_buf; /* buffer to hold file description text */
    /* Open the HDF file */
    file_id = Hopen("example.hdf", DFACC_READ, 0);
    /* Initialize the AN interface */
    an_id = ANstart(file_id);
```

```
    /* Get the annotation information */
    status = ANfileinfo(an_id, &n_file_labels, &n_file_descs,
            &n_object_labels, &n_object_descs);
    /* Print the number of each type of annotation */
    printf("number of file labels: %d\n", n_file_labels);
    printf("number of file descriptions: %d\n", n_file_descs);
    printf("number of object labels: %d\n", n_object_labels);
    printf("number of object descriptions: %d\n", n_object_descs);
    /* Select the first file label and the first file description */
    file_label_id = ANselect(an_id, 0, AN_FILE_LABEL);
    file_desc_id = ANselect(an_id, 0, AN_FILE_DESC);
    /* Get the length of the file label and file description */
    file_label_len = ANannlen(file_label_id);
    file_desc_len = ANannlen(file_desc_id);
    /* Allocate space to hold the annotation text */
    file_label_buf = malloc(file_label_len + 1);
    file_desc_buf = malloc(file_desc_len + 1);;
    /* Read the file label and file description */
    status = ANreadann(file_label_id, file_label_buf, file_label_len + 1);
    status = ANreadann(file_desc_id, file_desc_buf, file_desc_len + 1);
    /* Print the file label and file description */
    printf("file label: %s\n", file_label_buf);
    printf("file description: %s\n", file_desc_buf);
    /* Free memory space */
    free(file_label_buf);
    free(file_desc_buf);
    /* Terminate access to the annotations */
    status = ANendaccess(file_label_id);
    status = ANendaccess(file_desc_id);
    /* Terminate access to the AN interface, close the file */
    status = ANend(an_id);
    status = Hclose(file_id);
}
```

Fortran:

```
      program read_an
      integer DFACC_READ
      integer AN_FILE_LABEL
      integer AN_FILE_DESC
      parameter (DFACC_READ = 1,
     +        AN_FILE_LABEL = 2,
     +        AN_FILE_DESC = 3)
C     Function declaration
```

```
      integer hopen, hclose
      integer afstart, affileinfo, afselect, afannlen
      integer afreadann, afendaccess, afend
C     Variable declaration
      integer status
      integer file_id
      integer an_id
      integer n_file_labels, n_file_descs
      integer n_object_labels, n_object_descs
      integer file_label_id, file_desc_id
      integer file_label_len, file_desc_len
      character*256 file_label_buf, file_desc_buf
C     Open the HDF file
      file_id = hopen('example.hdf', DFACC_READ, 0)
C     Initialize the AN interface
      an_id = afstart(file_id)
C     Get the annotation information
      status = affileinfo(an_id, n_file_labels, n_file_descs,
     +        n_object_labels, n_object_descs)
C     Print the number of each type of annotation
      write(*,*) 'number of file labels: ', n_file_labels
      write(*,*) 'number of file descriptions: ', n_file_descs
      write(*,*) 'number of object labels: ', n_object_labels
      write(*,*) 'number of object descriptions: ', n_object_descs
C     Select the first file label and the first file description
      file_label_id = afselect(an_id, 0, AN_FILE_LABEL)
      file_desc_id = afselect(an_id, 0, AN_FILE_DESC)
C     Get the length of the file label and the file description
      file_label_len = afannlen(file_label_id)
      file_desc_len = afannlen(file_desc_len)
C     Read the file label and file description
      status = afreadann(file_label_id, file_label_buf, file_label_len+1)
      status = afreadann(file_desc_id, file_desc_buf, file_desc_len+1)
C     Print the file label and file description
      write(*,*) 'file label: ', file_label_buf(1:file_label_len)
      write(*,*) 'file description: ', file_desc_buf(1:file_desc_len)
C     Terminate access the annotations
      status = afendaccess(file_label_id)
      status = afendaccess(file_desc_id)
C     Terminate access to the AN interface, close the file
      status = afend(an_id)
      status = fclose(file_id)
      end
```

如下是本程序的输出结果：

```
number of file labels: 1
number of file descriptions: 1
number of object labels: 0
number of object descriptions: 0
file label: This is a file label
file description: This is a file description
```

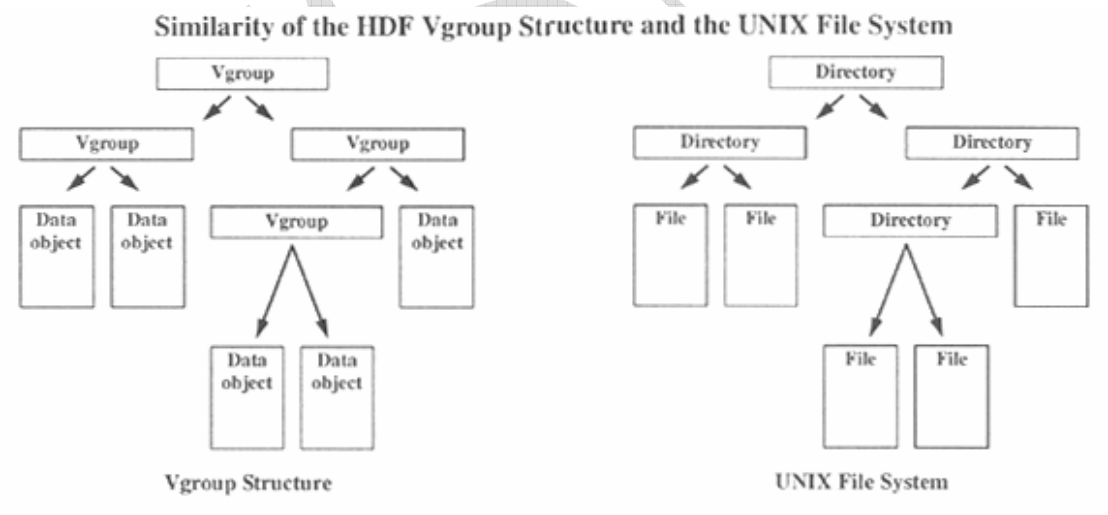
第七章 Vgroups（V API）

7.1 本章简介

本章描述 Vgroup 数据模式（V）和 V API。将介绍 V API 用于创建和处理 Vgroup 的接口例程。

7.2 Vgroup 数据模式

Vgroup 是一个关联相关数据对象的结构。Vgroup 的一般结构与 UNIX 文件系统的结构类似，Vgroup 可以包含另外的 Vgroup 或 HDF 数据对象的引用，如 UNIX 的目录可以包含子目录或文件一样（见图 7a）。属于 Vgroup 的数据对象通常称为 Vgroup 成员。



Vgroup 和 UNIX 文件系统的相似性

Vgroup 有一个必须的 Vgroup 名，和一个与之有关的可选项 Vgroup 类。在描述和分类属于 Vgroup 的数据对象时，要用到 Vgroup 名和类。例如，一个名为“Storm Tracking Data-5/11/94”的 Vdata 对象和另一个名为“Strom Tracking Data-6/23/94”的 Vdata 对象，同在一个名为“Storm Tracking Data-1994”的 Vgroup 下。假如这些数据是在 Alaska 的 Anchorage 收集的，类名就可以是“Anchorage Data”，特别是一些其它 Vgroup 含有从其它地方获得的暴雨路径数据。

7.3 V API

Vgroup 接口是由创建和存取 Vgroup 以及获取这些 Vgroup 信息和成员的例程组成的。

Steps or Sequence of function calls to access a Vgroup.

Function Call Steps	C	Fortran
1. Open the HDF file	Hopen	hopen
2. Initialize the Vgroup interface	Vstart	vfstart
3. Create or open a Vgroup	Vattach	vfatch
4. Perform the desired operations	Vaddtagref, Vdelete	vfadtr, vdelete
5. Terminate access to the Vgroup	Vdetach	vfdtch
6. Terminate access to the Vgroup interface	Vend	vfend
7. Close the HDF file	Hclose	hclose

表 7a　Vgroup 步骤或函数调用顺序

Vgroup 接口例程名的前缀为“V”（C 程序）或“vf”（FORTRAN 程序）。调用程序访问 Vgroup 时有下列步骤或函数调用顺序。

每个函数调用的详细信息见 HDF4 参考手册。

7.4 创建 Vgroup 和添加数据对象

执行完第三章到第六章的程序例子后，“example.hdf”文件业已创建了 5 个对象。这些对象是：GR 对象，SDS 对象，Vdata 对象，文件标签和文件描述。

在下列示例中，创建一个新的被称为“My Vgroup”的 Vgroup 对象，文件中已经存在的 GR 对象和 SDS 对象将被插入到这个 Vgroup 里。

在此例的 V 接口例程中，用 Vattach/vfatch 创建 Vgroup 对象，用 Vaddtagref/vfadtr 把 GR 和 SD 对象插入 Vgroup 里。对象被插入时，Vaddtargref/vfadtr 需要插入对象的标签和引用号。参见第 1.6 节对 HDF 数据对象标签和参考号的描述。

C:

```
/* write_Vgroup.c - create a Vgroup and add data objects */

#include "hdf.h"

main()
{
    int32 status,    /* status returned by function calls */
          file_id,   /* HDF file identifier */
          Vgroup_id, /* Vgroup identifier */
          gr_id,             /* GR interface identifier */
          sd_id,             /* SD interface identifier */
          rig_id,            /* raster image identifier */
          sds_id,            /* scientific data set identifier */
```

```
            rig_ref,    /* raster image reference number */
            sds_ref;    /* sds reference number */

    /* Open file "example.hdf" for writing */
    file_id = Hopen("example.hdf", DFACC_WRITE, 0);
    /* Initialize the V interface */
    status = Vstart(file_id);
    /* Create a new Vgroup */
    Vgroup_id = Vattach(file_id, -1, "w");
    /* Set Vgroup name as "My Vgroup" */
    status = Vsetname(Vgroup_id, "My Vgroup");
    /* Set Vgroup class as "Example" */
    status = Vsetclass(Vgroup_id, "Example");
    /* Initialize the GR and SD interfaces */
    gr_id = GRstart(file_id);
    sd_id = SDstart("example.hdf", DFACC_READ);
    /* Select the first image object and the first SDS object */
    rig_id = GRselect(gr_id, 0);
    sds_id = SDselect(sd_id, 0);
    /* Get the reference number of the image and the SDS */
    rig_ref = GRidtoref(rig_id);
    sds_ref = SDidtoref(sds_id);
    /* Add the image and the SDS to Vgroup */
    status = Vaddtagref(Vgroup_id, DFTAB_RIG, rig_ref);
    status = Vaddtagref(Vgroup_id, DFTAB_NDG, sds_ref);
    /* Terminate access to the image and to the SDS */
    status = GRendaccess(rig_id);
    status = SDendaccess(sds_id);
    /* Terminate access to the GR and SD interfaces */
    status = GRend(gr_id);
    status = SDend(sd_id);
    /* Terminate access to the Vgroup */
    status = Vdetach(Vgroup_id);
    /* Terminate access to the V interface and close the file */
    status = Vdetach(Vgroup_id);
    status = Hclose(file_id);
}
```

Fortran:

```
program write_Vgroup
C   Parameter declaration
    integer DFACC_WRITE
    integer DFACC_READ
```

```
      integer DFTAB_RIG
      integer DFTAB_NDG
      parameter (DFACC_WRITE = 2,
     +        DFACC_READ = 1,
     +        DFTAB_RIG = 306,
     +        DFTAB_NDG = 720)
C     Function declaration
      integer hopen, hclose
      integer vfstart, vfatch, vfsnam, vfscls, vfadtr, vfdtch, vfend
      integer mgstart, mgselct, mgid2ref, mgendac, mgend
      integer sfstart, sfselct, sfid2ref, sfendacc, sfend
C     Variable declaration
      integer status, file_id, Vgroup_id
      integer gr_id, rig_id, rig_ref
      integer sd_id, sds_id, sds_ref
C     Open file "example.hdf" for writing
      file_id = hopen("example.hdf", DFACC_WRITE, 0)
C     Initialize the V interface
      status = vfstart(file_id)
C     Create a new Vgroup
      Vgroup_id = vfatch(file_id, -1 'w')
C     Set Vgroup name as "My Vgroup"
      status = vfsnam(Vgroup_id, 'My Vgroup')
C     Initialize the GR and SD interfaces
      gr_id = mgstart(file_id)
      sd_id = sfstart('example.hdf', DFACC_READ)
C     Select the first image object and the first SDS object
      rig_id = mgselct(gr_id, 0)
      sds_id = sfselct(sd_id, 0)
C     Get the reference number of the image and the SDS
      rig_ref = mgid2ref(rig_id);
      sds_ref = sfid2ref(sds_id);
C     Add the image and the SDS to Vgroup
      status = vfsdtr(Vgroup_id, DFTAB_RIG, rig_ref)
      status = vfsdtr(Vgroup_id, DFTAB_NDG, sds_ref)
C     Terminate access to the image and to the SDS
      status = mgendac(rig_id)
      status = sfendacc(sds_id)
C     Terminate access to the GR and SD interfaces
      status = mgend(gr_id)
      status = sfend(sd_id)
C     Terminate access to the Vgroup
      status = vfdtch(Vgroup_id);
C     Terminate access to the V interface and close the file
```

```
status = vfend(file_id)
statue = hclose(file_id)
end
```

7.5 获取 Vgroup 的信息和删除数据对象

下例读取“example.hdf”文件，获取上面例子创建的 Vgroup 对象的信息。

在此例中，用 Vntagrefs/vfntr 获取 Vgroup 中数据对象的总数，用 Vgettagref/bfgttr 获取 Vgroup 每对成员的标记/引用号，用 Vdeletagref/vfdtr 从 Vgroup 分离出数据对象。

C:

```
/* read_Vgroup.c - get object information of a Vgroup and delete objects */
#include "hdf.h"
main()
{
    int32 status,    /* status returned by function calls */
          file_id,   /* HDF file identifier */
          vg_ref,            /* Vgroup reference number */
          vg_id,             /* Vgroup identifier */
          n_objects, /* number of data objects in a Vgroup */
          tag,               /* tag number of a data object */
          ref,               /* reference number of a data object */
          i;
    /* Open HDF file */
    file_id = Hopen("example.hdf", DFACC_WRITE, 0);
    /* Initialize the V interface */
    status = Vstart(file_id);
    /* Get the reference number of the first Vgroup */
    vg_ref = Vgetid(file_id, -1);
    /* Attach to the first Vgroup */
    vg_id = Vattach(file_id, vg_ref, "w");
    /* Get the total number of objects in the group */
    n_objects = Vntagrefs(vg_id);
    printf("%d objects found\n", n_objects);
    /* Print the tag and reference number for each object */
    for (i = 0; i < n_objects; i++)
    {
            /* Get the tag and reference number of each object */
            status = Vgettagref(vg_id, i, &tag, &ref)
            /* Print tag and reference number of each object */
            printf("index=%d, tag=%d, ref=%d", i, tag, ref);
            /* Delete this object from the Vgroup */
            status = Vdeletagref(vg_id, tag, ref);
    }
```

```
    /* Delete the first Vgroup from the file */
    status = Vdelete(file_id, vg_id);
    /* Terminate access to the Vgroup */
    status = Vdetach(vg_id);
    /* Terminate access to the V interface and close the file */
    status = Vend(file_id);
    status = Hclose(file_id);
}
```

Fortran:

```
      program read_Vgroup
C     Parameter declaration
      integer DFACC_WRITE
      parameter (DFACC_WRITE = 2)
C     Function declaration
      integer hopen, hclose
      integer vfstart, vfgid, vfatch, vntrc, vfgttr
      integer vfdtr, vdelete, vfdtch, vfend
C     Variable declaration
      integer status, file_id
      integer vg_ref, vg_id
      integer tag, ref, n_objects, i
C     Open HDF file
      file_id = hopen('example.hdf', DFACC_WRITE, 0)
C     Initialize the V interface
      status = vfstart(file_id)
C     Get the reference number of the first Vgroup
      vg_ref = vfgid(file_id, -1)
C     Attach to the first Vgroup
      vg_id = vfatch(file_id, vg_ref, 'w')
C     Get the total number of objects in the group
      n_objects = vntrc(vg_id)
      write(*,*) n_objects, ' objects found'
C     Print the tag and reference number for each object
      do 10 i = 1, n_objects
              status = vtgttr(vg_id, i, tag, ref)
              write(*,*) 'index=', i, ', tag=', tag, ', ref=', ref
C     Delete this object from the Vgroup
              status = vfdtr(vg_id, tag, ref)
      10 continue
C     Delete the first Vgroup from the file
      status = vdelete(file_id, vg_id)
C     Terminate access to the Vgroup
```

```
      status = vfdtch(vg_id)
C     Terminate access to the V interface and close the file
      status = vfend(file_id)
      status = hclose(file_id)
      end
```

如下为此程序的输出结果：

```
2 objects found
index=1, tag=306, reference=2
index=2, tag=720, reference=6
```

Vgroup 接口程序能提供比上述例子多得多的例程，每个 Vgroup 接口例程的详细描述请参见 HDF 用户手册和 HDF 参考手册。

第八章 HDF 命令行实用工具

8.1 本章简介

本章将介绍 HDF 命令行实用工具。在本章中，用户将有机会练习一些最有用的 HDF 实用命令行工具，如 hdp 和 vshow。

8.2 HDF 命令行实用工具介绍

HDF 软件包提供了一组命令行实用工具。HDF 命令行实用工具是在命令行提示符下执行一些小的应用程序。这些程序可使用户不用编写自己的程序即可执行普通操作。HDF 命令行工具分为三类：查询工具，转换工具和压缩工具。表 8a 列出了这些工具的名称和描述。假如已经安装了 HDF4.1r3 软件包，就可以在 HDF4.1r3/bin 子目录里找到这些程序。设置正确的路径后，能从任何子目录执行这些命令行工具。完整的 HDF 命令行工具列表可以在 http://hdf.ncsa.uiuc.edu/UG41r3_html/UG_BookTOC15.html 中找到。

Command line utitlites and a description.

Type		Name	Description
Inquiry		hdp	HDF dumper displays general information about the contents of an HDF file.
		hdfls	Displays tag and reference number of the data objects in an HDF file.
		vshow	Displays Vdata information in an HDF file.
		hdfed	Allows limited manipulation of data elements of an HDF file.
Conversion	Raw to HDF	fp2hdf	Converts 32–bit floating point data to scientific data set or 8-bit raster image.
		r8tohdf	Converts raw 8-bit images to HDF raster image sets.
		r24hdf8	Converts raw 24 bit images to HDF 8-bit raster images.
		paltohdf	Converts raw palettes to HDF format.
	HDF to Raw	hdftor8	Converts HDF raster image sets to raw format and separates them into image files and palette files.
		hdftopal	Converts palettes in HDF files to raw format and write them into separate files.
	HDF to HDF	ristosds	Converts several 8-bit raster images to one 3D scientific data set.
		hdf24hdf8	Converts 24–bit HDF raster images to 8-bit HDF raster images.
Compression		hdfcomp	Compresses 8-bit raster image from an HDF file, stores them into a new file.
		hdfpack	Condense HDF files by removing the empty spaces left by deleted objects.

表 8a HDF 命令行工具

8.3 HDF 查询工具

8.3.1 hdp

hdp（HDF dumper）工具是从指定的 HDF 文件中获取所有对象通用信息最为有用的工具。它可列出 HDF 文件在各层的详细内容。它还能倾印出文件中一个或多个特定对象的数据。hdf 提供一组命令，允许用户确定显示何种信息。下面例子展示怎样使用 hdf 获得 example.hdf 的信息，example.hdf 是由前面章节的程序创建的。如果没有创建你自己的 example.hdf，可以下载 example.hdf。

进入 example.hdf 文件所在的子目录，确保命令路径包含 HDF 工具目录，然后键入下列命令：

示例输出：hdp list 显示 HDF 文件的内容。

C:\HDF4.1r3\bin>hdp list example.hdf

```
File: example.hdf
File library version: Major= 4, Minor=1, Release=3
String=NCSA HDF Version 4.1 Release 3, May 1999
Version Descriptor : (tag 30)
    Ref nos: 1
File Identifier : (tag 100)
    Ref nos: 1
File Description : (tag 101)
    Ref nos: 1
```

```
Number type : (tag 106)
    Ref nos: 1 12
Image Dimensions : (tag 300)
    Ref nos: 1
Raster Image Data : (tag 302)
    Ref nos: 1
Raster Image Group : (tag 306)
    Ref nos: 1
SciData dimension record: (tag 701)
    Ref nos: 12
Scientific Data : (tag 702)
    Ref nos: 5
Numeric Data Group : (tag 720)
    Ref nos: 4
Vdata : (tag 1962)
    Ref nos: 6 8 10 15
Vdata Storage : (tag 1963)
    Ref nos: 6 8 10 15
Vgroup : (tag 1965)
    Ref nos: 2 3 7 9 11 13 14 16
```

示例输出：**hdp dumpsds** 显示 HDF 文件中科学数据集的内容

C:\HDF4.1r3\bin>hdp dumpsds example.hdf

```
File name: example.hdf
Variable Name = my SDS
    Index = 0
    Type= 64-bit floating point
    Ref. = 4
    Rank = 3
    Number of attributes = 0
    Dim0: Name=fakeDim0
    Size = 2
    Scale Type = number-type not set
    Number of attributes = 0
    Dim1: Name=fakeDim1
    Size = 5
    Scale Type = number-type not set
    Number of attributes = 0
    Dim2: Name=fakeDim2
    Size = 3
    Scale Type = number-type not set
    Number of attributes = 0
    Data :
    1.000000 2.000000 3.000000
```

```
    4.000000 5.000000 6.000000
    7.000000 8.000000 9.000000
    10.000000 11.000000 12.000000
    13.000000 14.000000 15.000000
    16.000000 17.000000 18.000000
    19.000000 20.000000 21.000000
    22.000000 23.000000 24.000000
    25.000000 26.000000 27.000000
    28.000000 29.000000 30.000000
```

示例输出：hdp dumpsds 显示 HDF 文件中科学数据集的内容。

C:\HDF4.1r3\bin>hdp dumpgr example.hdf

```
File name: example.hdf
    Image Name = My Image
    Index = 0
    Type= 8-bit unsigned integer
    width=10; height=8
    Ref. = 2
    ncomps = 1
    Number of attributes = 0
    Interlace= 0
    Data :
            0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
            20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
            37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
            54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
            71 72 73 74 75 76 77 78 79
```

示例输出：hdp dumpgr 显示 example.hdf 文件中 GR 对象的内容。

C:\HDF4.1r3\bin>hdp dumpvd –n "Simulation Data 1" example.hdf

File name: example.hdf

```
File Label #0: This is a file label
File description #0: This is a file description
Vdata: 3
    tag = 1962; reference = 15;
    number of records = 4; interlace = FULL_INTERLACE (0);
    fields = [X, Y, Temp];
    record size (in bytes) = 12;
    name = Simulation Data 1; class = 2D_Temperature_Grid;
    number of attributes = 0
- field index 0: [X], type=5, order=1
    number of attributes = 0
- field index 1: [Y], type=5, order=1
```

```
    number of attributes = 0
- field index 2: [Temp], type=5, order=1
    number of attributes = 0
Loc.Data
0  2.300000  1.500000  23.500000  3.400000  5.700000  8.030000
0.500000  3.500000  1.220000  1.800000  2.600000  0.000000
```

示例输出：hdp dumpvd 显示 HDF 文件中 Vdata 对象的内容。

```
C:\HDF4.1r3\bin>hdp dumpvd -n "Simulation Data 1" example.hdf
File name: example.hdf
File Label #0: This is a file label
File description #0: This is a file description
Vdata: 3
    tag = 1962; reference = 15;
    number of records = 4; interlace = FULL_INTERLACE (0);
    fields = [X, Y, Temp];
    record size (in bytes) = 12;
    name = Simulation Data 1; class = 2D_Temperature_Grid;
    number of attributes = 0
- field index 0: [X], type=5, order=1
    number of attributes = 0
- field index 1: [Y], type=5, order=1
    number of attributes = 0
- field index 2: [Temp], type=5, order=1
    number of attributes = 0
Loc.Data
0  2.300000  1.500000  23.500000  3.400000  5.700000  8.030000
0.500000  3.500000  1.220000  1.800000  2.600000  0.000000
```

注意：命令“vshow example.hdf +”可用来查询“example.hdf”文件中数据对象名。

示例输出：hdp dumpvg 显示 HDF 文件中 Vgroups 的内容。

```
C:\HDF4.1r3\bin>hdp dumpvg -n "My Vgroup" example.hdf

File name: example.hdf
File Label #0: This is a file label
File description #0: This is a file description
Vgroup:7
    tag = 1965; reference = 16;
    name = My Vgroup; class = Example;
    number of entries = 2;
    number of attributes = 0
```

```
Entries:-
    #0 (Raster Image Group)
            tag = 306; reference = 2;
    #1 (Numeric Data Group)
            tag = 720; reference = 4;
```

注意：命令“vshow example.hdf +”可用来找出存于“example.hdf”文件中数据对象的名称。

注意：每个 hdp 命令都提供一个可选标志。完整信息请参见 HDF 用户指南。

8.3.2 hdfls

hdfls 工具列出 HDF 文件中每个数据对象的标记和引用号。这个命令行与 hdp list 命令的功能类似。

示例输出：hdfls 工具

```
C:\HDF4.1r3\bin>hdfls example.hdf
example.hdf:
File library version: Major= 4, Minor=1, Release=3
String=NCSA HDF Version 4.1 Release 3, May 1999
Version Descriptor    : (tag 30)
            Ref nos: 1
File Identifier              : (tag 100)
            Ref nos: 1
File Description      : (tag 101)
            Ref nos: 1
Number type           : (tag 106)
            Ref nos: 1 12
Image Dimensions      : (tag 300)
            Ref nos: 1
Raster Image Data     : (tag 302)
            Ref nos: 1
Raster Image Group    : (tag 306)
            Ref nos: 1
SciData dimension record      : (tag 701)
            Ref nos: 12
Scientific Data               : (tag 702)
            Ref nos: 5
Numeric Data Group    : (tag 720)
            Ref nos: 4
Vdata         : (tag 1962)
            Ref nos: 6 8 10 15
Vdata Storage                 : (tag 1963)
            Ref nos: 6 8 10 15
```

```
Vgroup                : (tag 1965)
            Ref nos: 2 3 7 9 11 13 14 16
```

8.3.3 vshow

vshow 工具显示 HDF 文件中 Vdata 对象的信息。例如，下列命令显示 example.hdf 文件中存于 Vdata 对象里的所有数据记录。

示例输出：vshow 工具

```
C:\HDF4.1r3\bin> vshow example.hdf -
FULL DUMP
FILE: example.hdf
Lone Vdatas:
L vs:15 <1962/15> nv=4 i=0 fld [X,Y,Temp] vsize=12 (Simulation Data 1
{2D_Temperature_Grid})
0: fld [X], type=5, order=1
1: fld [Y], type=5, order=1
2: fld [Temp], type=5, order=1
2.300000  1.500000  23.500000
3.400000  5.700000  8.030000
0.500000  3.500000  1.220000
1.800000  2.600000  0.000000
 0 attributes.
```

8.3.4 hdfed

HDF 编辑工具 hdfed 是一个针对 HDF 文件的行编辑器。允许有经验的 HDF 用户灵活处理 HDF 文件的元素。这些处理功能包括：

- 选择组并显示它们的信息
- 倾印组信息至输出文件
- 把组数据写入输出文件
- 从 HDF 文件里删除组
- 把组插入 HDF 文件
- 替换 HDF 文件中的要素
- 编辑 HDF 文件中任何要素的标记和描述符

hdfed 工具是为在数据要素层处理 HDF 文件的用户而设计的，它并非为 HDF 文件提供一个高级显示工具。为了有效使用 hdfed，需要对 HDF 格式有深入的了解，因此，建议那些不熟悉 HDF 低级别组件的用户不要使用它。完整介绍请参见 HDF 用户手册。

8.4 HDF 数据格式转换工具

8.4.1 原始数据到 HDF 的转换

工具有：r8tohdf，r24hdf8， paltohdf 和 fp2hdf。r8tohdf 工具把一组原始光栅图像转换为 HDF 8 比特光栅图像格式，并把它们写入文件。命令行类似如下：

r8tohdf rowscols 8-bit_raster.hdf raster_file.raw [-p palette_file.raw]

原始调色板文件是可选项。假如 raster.raw 文件为 256 x 512 的原始光栅图像。假定名为 palette.raw 的调色板以原始形式存于文件。如要创建一个光栅图像 HDF 文件，则需使用命令行：

r8tohdf256512raster_set.hdf-p palette_file.hdfraster.raw

r24hdf8 工具量化原始的 RGB 24 比特光栅图像，创建一个带有 256 色调色板的 8 比特图像，并把调色板和 8 比特的光栅图像存于 HDF 文件中。命令行句法为：

r24hdf8colsrowsraster24_file.raw raster8_file.hdf

paltohdf 工具把一个原始调色板文件转化为 HDF 格式。原始调色板文件有 768 字节，以下列顺序组织：256 红色值，256 绿色值和 256 兰色值。一旦调色板被转换成 HDF 格式，调色板数据就以 RGBRGB…RGB 的形式交替。把原始调色板转换为 HDF 调色板，执行类似于下面的命令：

paltohdf palette_file.raw palette_file.hdf

fp2hdf 工具把原 32 比特或 64 比特数据转换为一个 HDF 文件，此文件包含一个 8 比特的光栅图像数据集，一个 32 双浮点科学数据集，或两者都有。输入数据可以来自于文本文件、二进制文件或 HDF 文件。若转换包含 2 维数据的文本文件，数据必须为下列格式：

Format	Planes	Rows	Columns	Max. Data	Min. Data	Row Scales	Column Scales	Data
Text	1	10	20	0	255	row1 row2...row10	col1 col2...col20	data1 data2...data200

作为一个怎样使用 fp2hdf 的示例，考虑一个包含原始 32 比特浮点数据的文件。把该原始文件转换为一个包含 8 比特光栅图像和科学数据集的 HDF 文件，命令行操作如下：

fp2hdf float32.raw –o float32.hdf –r –f

-r 指示创建光栅数据集，-f 表示创建科学数据集。

8.4.2 把 HDF 转换为原始数据

工具：hdftor8 和 hdftopal。Hdftor8 工具从 HDF 文件中提取光栅图像和调色板，并把它们存于单独的原始数据文件中。例如，一个名为 input.hdf 的文件含有 3 个 512x256 光栅图像和 3 个调色板。命令行类似下面：

hdftor8 input.hdf

hdftor8 input.hdf 创建 6 个文件，名为 img001–512.256、img002–512.256、img003–512.256、pal001、pal002、pal003。hdftopal 工具把 HDF 文件中的调色板转换为原始数据文件。原始数据文件有 768 字节，第一个 256 字节代表红色强度值，第二个 256 字节代表绿色强度值，第三个 256 字节代表兰色强度值。这个工具执行 paltohdf 工具的转换操作。命令行类似如下：

hdftopal HDF–format–palette–filename raw–format–palette–filename

8.4.3 HDF 到 HDF 的转换工具

irstosds 和 hdf24hdf8。ristosds 工具把一组连续的 HDF 光栅图像文件转换为单个的 HDF 科学数据集。生成的科学数据集是一个 3 维数组，宽和高与光栅图像的维数相等。因此，序列光栅图像必须有相同的维数。数组的第三维，或称长度，由转换中所包含的图像总数确定。假如图像中包含调色板，这个调色板必须在第一个 HDF 输入文件中。这些图像只能与一个调色板关联，处理完第一个调色板后，该实用工具将忽略它所遇到的任何其它别的调色板数据。命令行语法为：

ristosds image_file_1.hdf image_file_2.hdf… image_file_n.hdf–osds_file.hdf

hdf24hdf8 工具把 24 比特的光栅图像数值量化为一个 8 比特的具有 256 色调色板的光栅图像，并把这个调色板和 8 比特光栅图像存于一个 HDF 文件里。命令行语法为：

hdf24hdf824–bit_image_file.hdf8–bit_image_file.hdf

8.5 HDF 数据压缩工具

hdfpack 和 hdfcomp。Hdfpack 工具通过移去删除数据元素后所留下的空格来压缩 HDF 文件，这些空格是数据要素删除后留下的。通过从解包文件中读取现存的数据，然后把它们写入一个新的打包文件中。命令行如下：

hdfpack unpacked_file.hdfpacked_file.hdf

hdfpack 工具还提供一些附加选项，以定制 HDF 文件的压缩方式。有关这些选项的详细介绍见 HDF 用户手册。

Hdfcomp 工具把 8 比特的光栅图像读入一个或多个 HDF 文件里，对它们进行压缩并把压缩图像存于一个新的 HDF 文件里。假如输出的 HDF 文件已经存在，这些压缩的图像将附

在它后面。命令行语法如下：

hdfcomp output.hdf[–c| –r| –i] input_1.hdf [–c| –r| –i] input_2.hdf … [–c| –r| –i] input_n.hdf

这里-c 表示通过 run-length encoding 压缩，-I 表示使用 IMCOMP 算法压缩，-r（缺省）保留图像为非压缩状态。

第九章 使用 JHV 浏览 HDF 文件

9.1 本章简介

本章介绍 HDF 可视化工具 JHV（Java HDF Viewer）。用户可以把 JHV 下载和安装在自己的机器上，通过它可浏览示例 HDF 文件的内容并显示数据对象。

9.2 什么是 JHV

有许多软件工具可以用来显示和分析 HDF 文件。有些是可以从公众服务部门获得（免费），有些则是由商业公司开发的。

JHV 是基于 Java 的工具，易于显示 HDF 文件的内容。JHV 是由 NCSA 开发的，可以免费提供给公众。JHV 允许用户浏览完整的 HDF 文件。从 HDF 文件的最高层对象开始显示，JHV 允许用户向下通过层次，并在文件的数据对象中导航。只有在数据对象被选择时，其内容在会被装载，这时才提供接口和有权访问 HDF 文件。HDF 文件的结构表现为树状结构，其数据组和数据对象由通常的文件夹和图标表示。用户通过很方便地打开或关闭文件夹来对 HDF 文件的层状结构进行操作。

JHV 支持的 HDF 版本是 HDF4，而不是 HDF5。关于 HDF4 和 HDF5 的区别请参见 1.7 节。关于 HDF5 更详细的内容，参见 NCSA 的 HDF5 工具页。

9.3 获得和安装 JHV

JHVNCSA 是在基于 Java 的产品页面（http://hdf.ncsa.uiuc.edu/java-hdf-html/download.html）上发布的。JHV 需要 Java 运行时的环境（JER），JER 也可随同 JHV 应用软件一起下载。JHV 支持的 JRE 的版本为 JRE1.1.6、JRE1.1.7 和 JRE1.1.8。

注意：JHV 与 JDK1.2 一同使用有点问题，这是因为 JFC/Swing 用的是 JDK1.2 或早期的 Java 图形工具包（AWT）有区别而导致的。这个问题将会在将来的 JHV 版本中解决。

对于 WINDOWS9x/NT 系统，下载 zip 文件，解包到一个临时目录里，然后执行 Setup.exe 文件。按照安装指南，提示时给出想要 JHV 安装的路径。假如要从要求的 JRE 包中安装 JHV，还需要给 JRE 指定路径。

假如从一个存在的 JRE 包里安装 JHV，安装步骤将会要求指定 JRE 存在的位置。

发布的 JHV 可以由两种方式得到：

- ✧ 完整的 JRE 捆绑包。此软件包可以独立于其它软件而安装和运行。如果用户的机器没有安装 JRE，或不被 JHV 支持，则须下载此软件包。

✧ JHV 要求的软件包，当与安装的 JRE 兼容时，能在任何系统上运行。

启用 JHV 时，点击 Start，*Programs –> NCSA JHV2.5*，然后点击 click *NCSA JHV2.5*，或切换到 *ncsa\jhv\bin* 并且点击 jhv.bat。

对于 UNIX 系统，下载和执行自安装 SHELL 程序。对于完整的 JRE 捆绑包，只须如此做。对于要求的 JRE 包，用户需要设置如下的环境变量，

1. CLASSPATH — add the path to the NCSA JHI/JHV classes,
e.g. setenv CLASSPATH/usr/local/jhv/lib/classes.zip:$CLASSPATH

2. LD_LIBRARY_PATH — add the path to the NCSA JHI/JHV libraries,
e.g. setenv LD_LIBRARY_PATH/usr/local/jhv/lib:$LD_LIBRARY_PATH

3. PATH — add the path to the NCSA JHI/JHV binaries,
e.g. setenv PATH /usr/local/jhv/bin:$PATH

启动 JHV 时，在命令行键入 jhv 即可。

9.4 显示 HDF 对象树

正如前面章节所学的，使用 Vgroups 可以把 HDF 数据对象组织成一个树状结构。看一个 JHV 的例子，启动 JHV 应用程序，在 File 菜单中选择 Open Local File，在 ncsa/jhv/samples 子目录里选择 annras.hdf 文件。文件结构就在 HDF 的树状窗口中显示出来。数据对象显示为图标，HDF Vgroups 显示为文件夹，它可以包含对象和其它 Vgroups。这些图标的意义见图 9b。文件层的操作是通过选择文件夹来“open”一个 Vgroup，显示由 Vgroup 包含的对象。当选择了一个对象后，数据就恰当地显示在预览窗口中，图 9a 显示了 annras.hdf 文件的 HDF 树状结构、有一些打开的文件夹和几个可视的对象。在图 9a 里，一个图像对象被选择并被显示出。

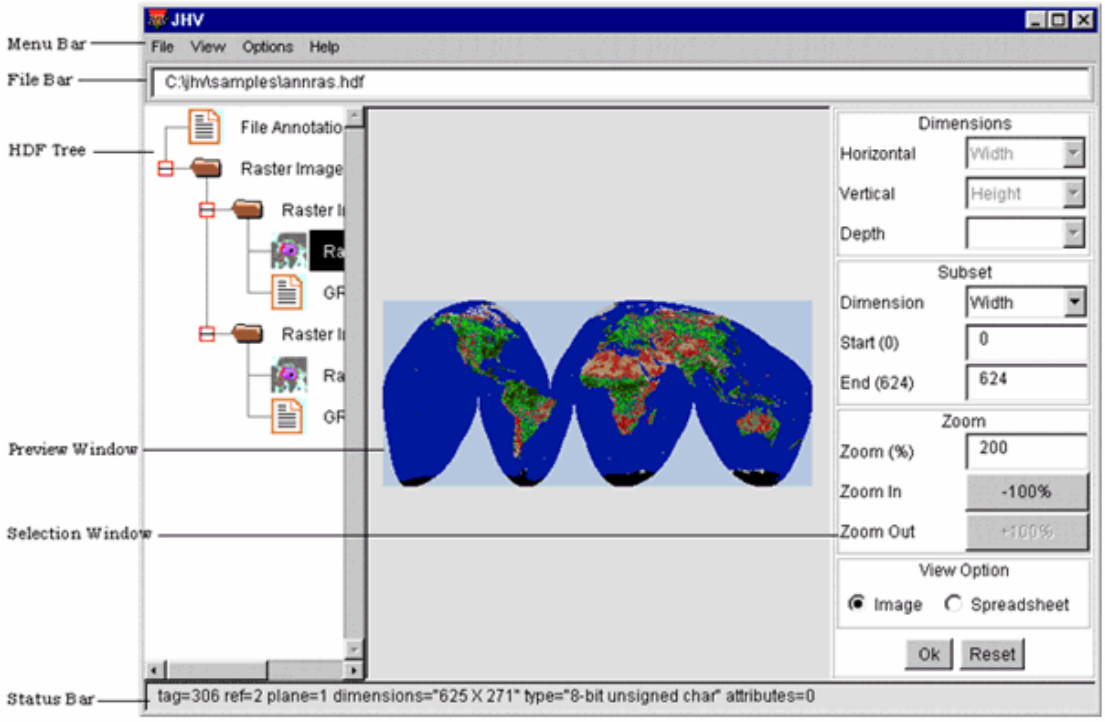


图 9a JHV 主窗口

 文件和数据对象注解（File and Data Object Annotation）

 光栅图像 （Raster Image）

 科学数据集（Scientific Data Set）

 虚拟数据（Vdatas）

 虚拟组数据（Vgroups）

图 9b JHV 图标

9.5 显示文件和数据对象注解

如第六章所描述的，HDF 注解是文本字符串，用来描述一个 HDF 文件或该文件所包含的任何数据要素。注解来源于两种形式：标签号和描述符。标签为短注释，用来把诸如标题或时间标志之类的字串分配给一个文件或它后面的数据对象。描述符是长信息，它们通常含有非常广泛的信息，如原程序模型或数学公式。

两种注解类型：文件注释和数据对象注释。文件注释描述整个 HDF 文件。数据对象注解描述文件里特定的数据对象。文件和数据对象注释可以是标号或描述符。因此，一个注解可以是一个文件标签，一个文件描述符，一个对象标签，或一个对象描述符。尽管属性集与数据有关，但属性集在 HDF 树中却显示为独立对象。JHV 用与注解同样的方式显示属性集。JHV 显示附加给文件或对象的注解。当选择显示一注解时，JHV 就在预览窗口中显示注解文本。图 9c 是一个怎样显示注解的例子。

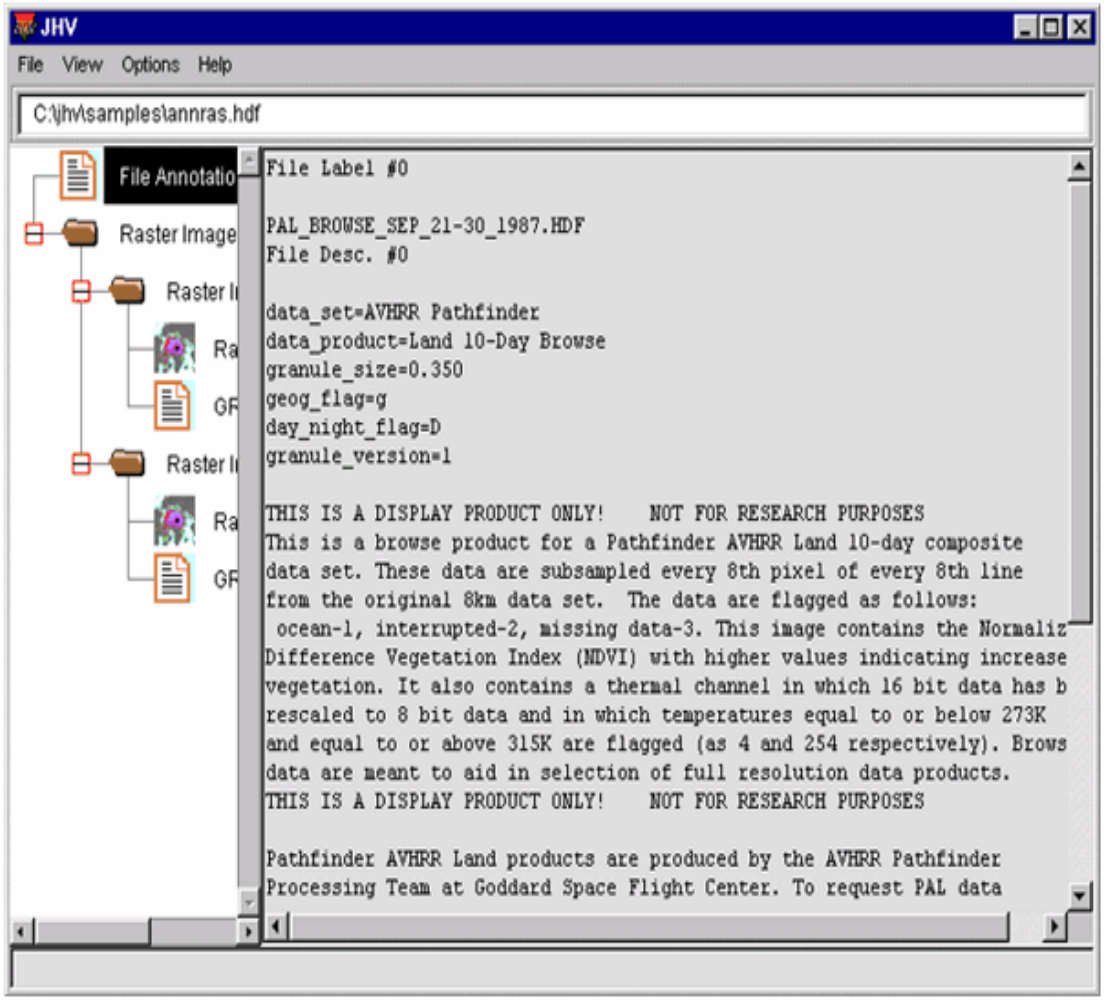


图 9c　显示文件注解

9.6 显示虚拟数据 Vdatas

HDF 的 Vdata 是一个记录集，其值存于定长域中。因此 Vdata 实质上是一个表格，表中每行有相同的结构，且每列含有相同类型的数据。下面是一个显示 Vdata 的例子。

1. 打开文件 ncsa/jhv/samples/sdsVdata.hdf，文件结构显示在左边
2. 打开 Vdata(lone) 文件夹
3. 打开具有混合数据类型的 Vdata 文件夹
4. 单击 Vdata 表：与 Vdata 结合的类型图标，弹出带有混合型图标的 Vdata，如下面的电子数据表窗口（图 9d）

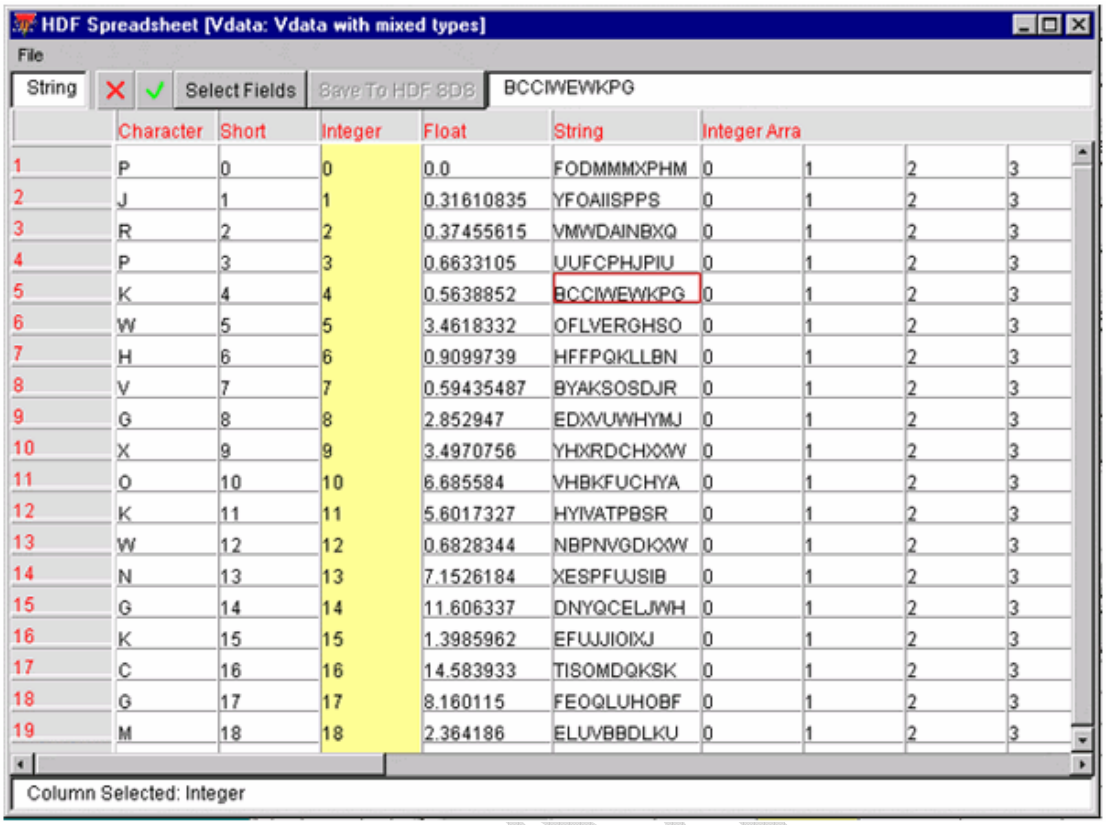


图 9d　Vdata 的电子数据表窗口

单击“Select Fields”按钮，打开“Select Vdata”窗口（图 9e）。此窗口允许用户选择要显示的域（列），也可以让用户设置要显示的记录范围。

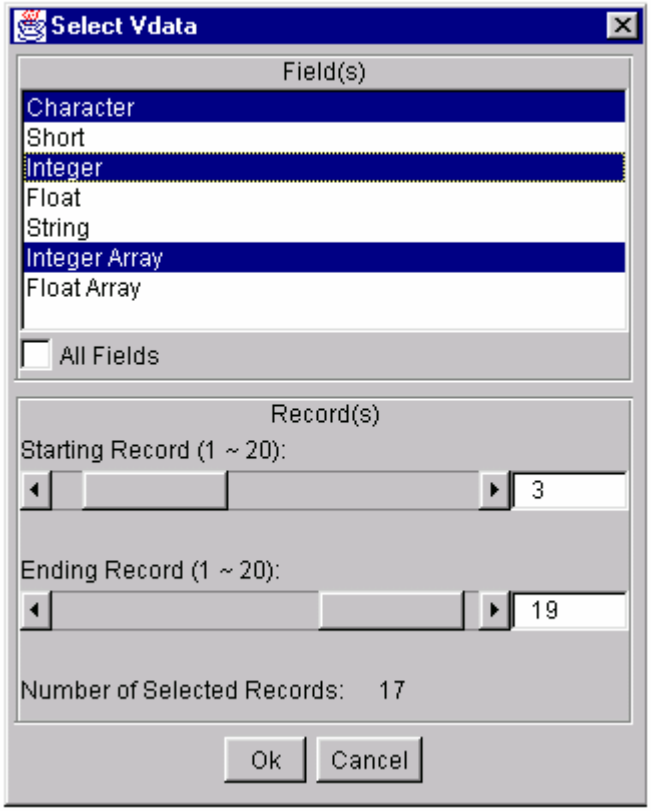


图 9e　选定 Vdata 窗口

9.7 显示光栅图像

JHV 可以让用户显示和处理 8 比特和 24 比特的光栅图像。图 9a 显示了怎样从一个 HDF 文件中选定一个图像对象。显示实际大小的图像，点选主窗口右下部分的“Image”单选钮，然后点击“OK”按钮，图像窗口就被打开。图 9f 是 JHV 的图像窗口。

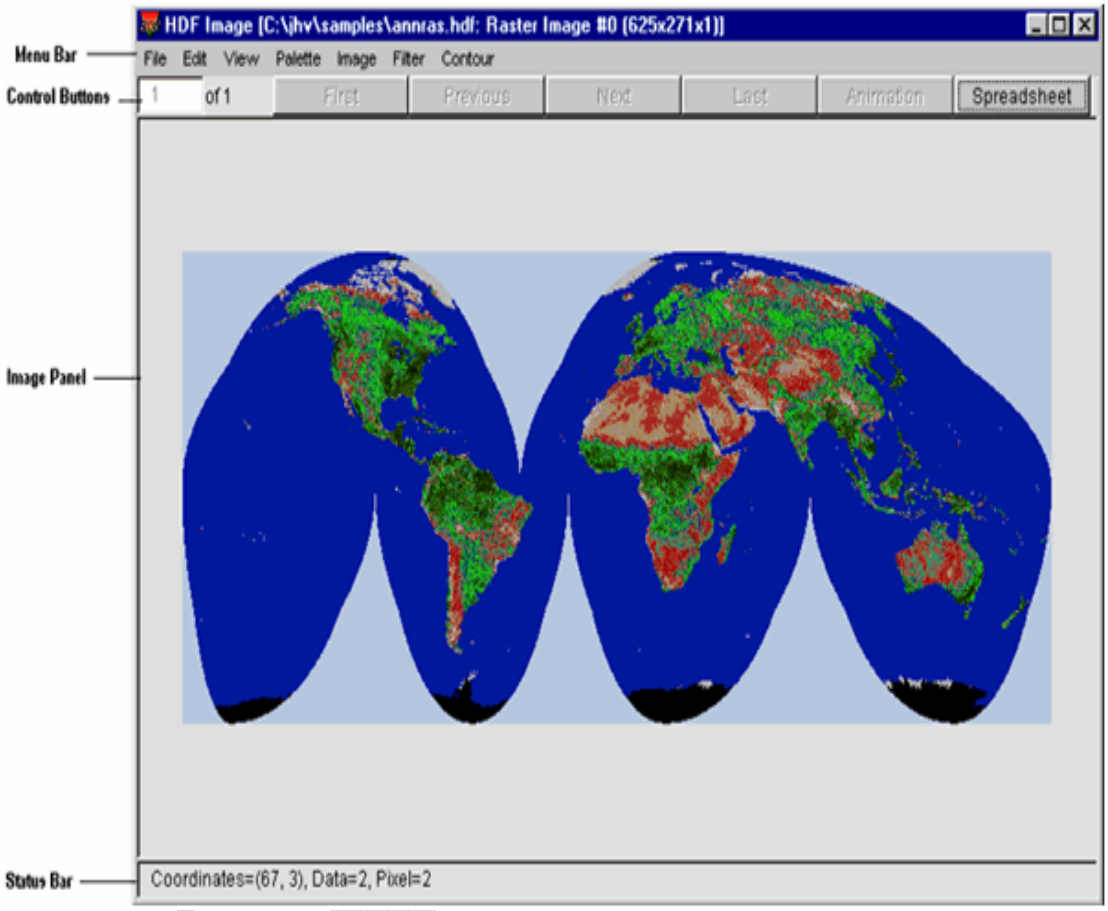


图 9f　JHV 图像窗口

JHV 可以让用户完成许多标准图像处理算法，类似于通常的图像编辑器所提供的。这些算法包括平滑（smooth），锐化（sharpen），增噪（add noise），查找边缘（find edge），翻转（flip），和凸显（emboss）等。用户何以改变图像的调色板，或使用调色板编辑器创建一个新的调色板（见图 9g）。

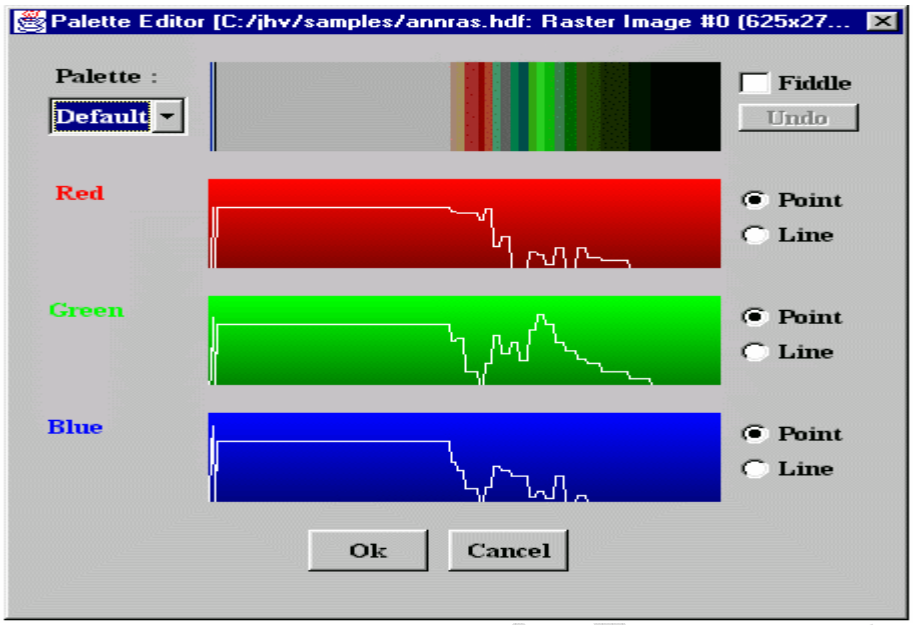


图 9g　调色板编辑器

9.8 显示科学数据集

目睹 SDS 的一个可视化例子，请打开示例文件“*sdsVdata.hdf*”，然后打开“Vgroup SDS Array”并选定数据对象“3D float array”。该 3D 数组的第一帧图像随即在预览窗口中显示出来（见图 9h）。

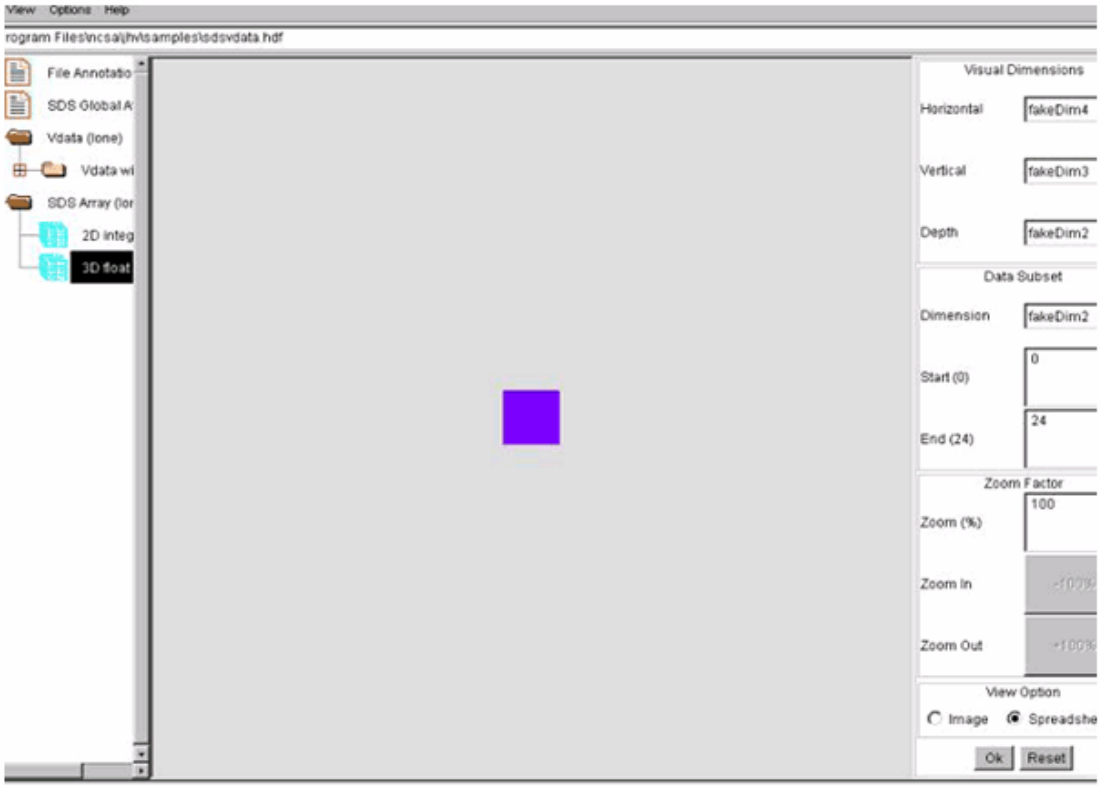


图 9h　显示一个 SDS 对象

点选右下角的“Image”广单选框，并单击“OK”即可调出图像窗口。在此图像窗口里，使用“Next”和“Last”按钮，逐幅浏览 3D 数组。单击“Animation”按钮，将弹出一个动画窗口。单击“Select All”按钮，点亮这个 3D 数组的所有图像画面，用滑动条设置动画速度，然后单击“Forward”按钮。播放 SDS 数组所有图像帧的动画（见图 9i）。

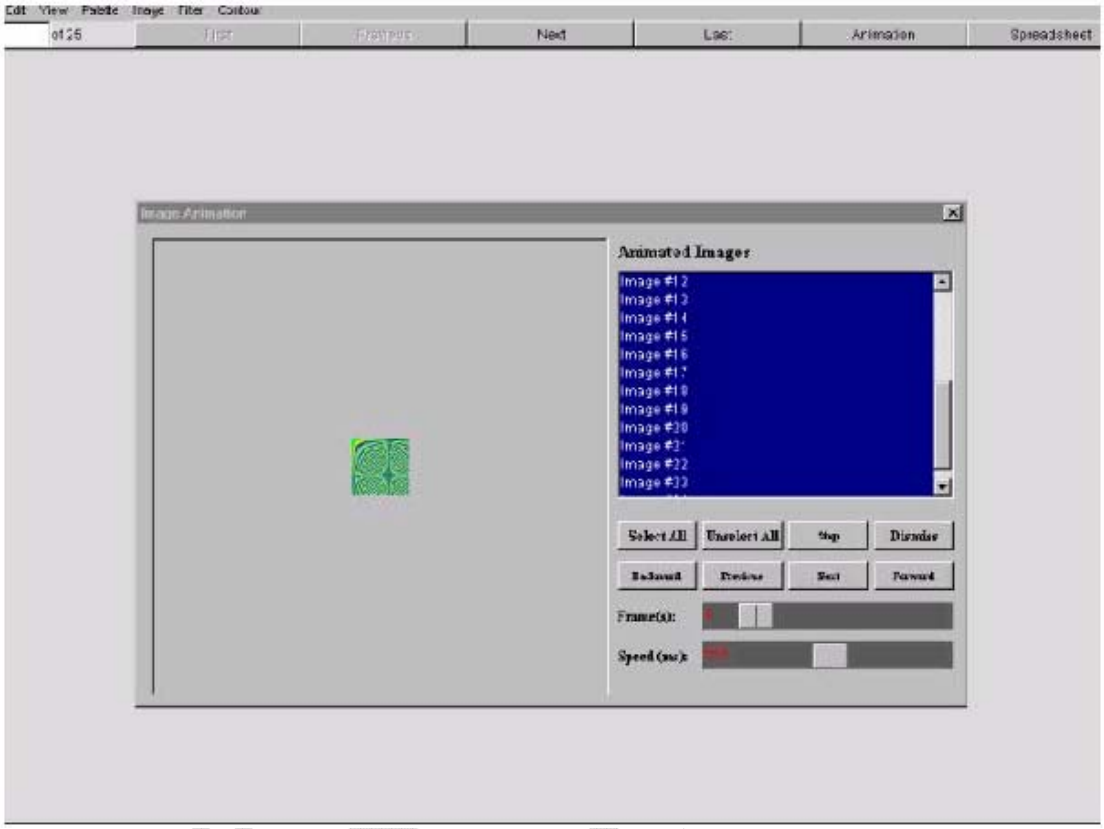


图 9i　显示作为动画的 3D SDS