HDF5.0 使用简介

NSMC

目 录

1、介绍（Introduction）.....1
2、HDF5 文件组织（File Organization）.....1
3、HDF5 应用程序接口（API）.....2
4、创建 HDF5 文件（Creating an HDF5 File）.....3
4.1 什么是 HDF5 文件?.....3
4.2 程序例子（Programming Example）.....4
4.2.1 描述（Description）.....4
4.2.2 备注（Remarks）.....5
4.2.3 文件内容（File Contents）.....6
4.2.4 DDL 中的文件定义（File Definition in DDL）.....7
5、创建数据集（Creating a Dataset）.....7
5.1 什么是数据集（What is a Dataset）.....7
5.2 数据类型（Datatypes）.....7
5.3 数据集和数据空间（Datasets and Dataspaces）.....9
5.4 数据集创建特性列表（Dataset Creation Property Lists）.....9
5.2 程序例子（Programming Example）.....10
5.2.1 描述（Description）.....10
5.2.2 备注（Remarks）.....11
5.2.3 文件内容（File Contents）.....12
5.2.4 DDL 中的数据集定义（Dataset Definition in DDL）.....13
6、创建一个群组（Creating a Group）.....14
6.1 什么是群组（What is a Group）?.....14
6.2 程序例子（Programming Example）.....15
6.2.1 描述（Description）.....15
6.2.2 备注（Remarks）.....15
6.2.3 文件内容（File Contents）.....16
7、创建属性（Creating an Attribute）.....17

7.1 什么是属性（Attribute）？.....17
7.1.1 创建一个属性（Creating an attribute）.....17
7.1.2 读/写属性（Reading/Writing an attribute）.....18
7.2 编程例子（Programming Example）.....18
7.2.1 描述（Description）.....18
7.2.2 备注（Remarks）.....19
7.2.3 文件内容（File Contents）.....21
7.2.4 DDL 中的属性定义（Attribute Definition in DDL）.....22
8、读出和写入数据集（Reading from and Writing to a Dataset）.....22
8.1 读出和写入数据集（Reading from and Writing to a Dataset）.....22
8.2 编程例子（Programming Example）.....23
8.2.1 描述（Description）.....23
8.2.2 备注（Remarks）.....24
8.2.3 文件内容（File Contents）.....27
9、复合数据类型（Compound Datatypes）.....28
9.1 创建复合数据类型（Creating Compound Datatypes）.....28
9.2 程序例子（Programming Example）.....29
9.2.1 描述（Description）.....29
9.2.2 备注（Remarks）.....29
9.2.3 文件内容（File Contents）.....30

NSMC

说明： 本教材不包含编译例程所需的软件。

Introductory Topics	Advanced Topics	Parallel HDF5	High Level APIs
Introduction	**Property Lists**	**Design Overview**	
HDF5 File Organization	**Compound Datatypes**	**Parallel Programming**	
The HDF5 API	**Hyperslab Selection**	**Create/Access File**	
Create an HDF5 File	**Point Selection**	**Create/Access Dataset**	**Other**
Create Dataset	**References to Objects**	**Write/Read Hyperslabs**	**Tutorial Examples**
Dataset Read/Write	**References to Regions**	**- by Contiguous Hyperslab**	**Utilities: h5ls/h5dump**
Create Attribute	**Extendible Datasets**	**- by Regularly Spaced Data**	**References**
Create Group	**Mounting Files**	**- by Pattern**	
Create Group - Abs/Rel	**Group Iteration**	**- by Chunk**	
Create Dataset in Group			
[Questions] [Answers]			

NSMC

1、介绍（INTRODUCTION）

欢迎使用由 HDF 用户支持组提供的 HDF5 使用教材

HDF5 是用于存储科学数据的一种文件格式和库文件。它被设计并实现满足科学数据存储不断增加和数据处理不断变化的需求，为了充分利用当今计算机系统的能力和特点，克服 HDF4.x 的不足。HDF5 有一个强大和灵活的数据模块，支持管理的文件大于 2 GB（HDF4.x 管理文件的极限），并且还支持并行 I/O。设计时考虑了安全线程并将在不久的将来实现此功能。为了便于简单了解 HDF5 的数据模式、库函数和工具，请参看存放于 URL 地址（http://hdf.ncsa.uiuc.edu/HDF5/papers/HDF5_overview/index.htm）里的幻灯片。

本使用教材涵盖了基本的 HDF5 数据对象和文件结构，HDF5 程序模块、创建和修改数据对象的 API 功能。还将介绍用于存取 HDF5 文件的一些有用的工具。

本教材使用的程序例子以及编译它们的 Makefile 在 ./examples/子目录里。为了使用 Makefile 文件,用户也许不得不编辑和更新编译器和编译器选项，以及发布的 HDF5 的目录路径。Java 程序例子在./examples/ 目录下的名为 java/子目录里。Java/目录有一个 Makefile 文件和运行 java 程序的脚本文件（shell scripts）。

对于其它 HDF5 程序的程序例子，请参看 References 里的指示标记。

希望这个程序例子和用法说明入门能帮助用户很方便地使用 HDF5。

任何意见和建议直接发给：hdfhelp@ncsa.uiuc.edu.

2、HDF5 文件组织（FILE ORGANIZATION）

一个 HDF5 文件就是一个由两种基本数据对象（groups and datasets）存放多种科学数据的容器：

- HDF5 group：包含 0 个或多个 HDF5 对象以及支持元数据（metadata）的一个群组结构。
- HDF5 dataset：数据元素的一个多维数组以及支持元数据（metadata）

任何 HDF5 的群组或数据集或许都有一个对应的属性列表。HDF5 属性是一个用户自定义的 HDF5 结构，能为 HDF5 对象提供附加信息。

使用群组和数据集时在许多方面类似于使用 UNIX 的目录和文件。HDF5 文件里的对象经常通过它的绝对路径来引用。

/ signifies the root group.
/foo signifies a member of the root group called foo.
/foo/zoo signifies a member of the group foo, which in turn is a member of the root group.

3、HDF5 应用程序接口（API）

HDF5 函数库提供几个应用程序接口（API）。这些 API 提供用于创建、存取、处理 HDF5 文件和对象的例程。

库函数本身是由 C 程序实现的。为了便于 FORTRAN90 和 Java 程序员的工作，HDF5 包装的外壳函数也已经用这些语言开发出来了。到写此用户教材时，用 C++包装的外壳函数正在开发中。本教材只涉及 C 和 FORTRAN 的外壳包装函数。

HDF5 库中所有 C 的例程都有一个前缀形式 H5*开始，这里的*是一个或两个大写字母，表明有关函数操作对象的类型。而 FORTRAN 外壳包装是以子程序由 h5 开始和_f 结束的。API 列表如下：

API	描述
H5	库函数：H5 函数的通用目的
H5A	注解(Annotation)接口：属性存取(access)和操作例程
H5D	数据集接口：数据集存取和操作接口
H5E	错误接口：错误处理例程
H5F	文件接口：文件存取例程
H5G	群组接口：群组创建和运行例程
H5I	标识号接口：标识号例程
H5P	特性（Property）列表接口：对象特性列表操作例程
H5R	引用接口：引用例程
H5S	数据大小接口：数据大小定义和存取例程
H5T	数据类型接口：数据类型创建和操作例程
H5Z	压缩接口：压缩例程

4、创建 HDF5 文件（CREATING AN HDF5 FILE）

4.1 什么是 HDF5 文件?

HDF5 是一个含有科学数据和支持元数据的二进制文件。HDF5 文件存储对象的基本类型，即群组和数据集，将在本教材的其它章节讨论。

要创建一个文件，应用程序必须指定一个文件名、文件存取模式、文件创建特性列表、和文件存取特性列表。

- **文件存取模式（File access mode）：**：
  当创建一个文件时，假如此文件已经存在，文件存取模式就会指定要发生的动作：
  - H5F_ACC_TRUNC 说明如果此文件已经存在，当前的内容将被删除以便应用程序可以用新数据重新写此文件。
  - H5F_ACC_EXCL 说明如果此文件存在，打开则会失败。
  - 如果此文件不存在，则文件的存取参数被忽略。
  - 在所有情况下，对于一个成功创建的文件，应用程序都可以对这个文件进行读写存取。

  注意对于打开已经存在的文件有两种不同的存取模式：

  - H5F_ACC_RDONLY 说明应用程序只有读取而没有写入任何数据的权利。
  - H5F_ACC_RDWR 说明应用程序有读写的权利。

  更详细的信息请参见《HDF 用户指南》里的 The File Interface (H5F) 小节和《HDF5 参考手册》中的 H5F: File Interface 小节。

- **文件创建特性列表（File creation property list）：**
  文件创建特性列表常被用于控制文件的元数据。文件的元数据包含有关用户块的大小、HDF5 库使用的不同文件数据结构大小等。此教材中，缺省的文件创建特性列表是 H5P_DEFAULT。

  用户块（user-block）是被 HDF5 库忽略的位于文件开始的固定长度的数据块。用户块可以用于存放任何数据或对应用有用的信息。

  更详细的内容请参见《HDF 用户指南》中的 The File Interface (H5F) 小节。

- **文件存取特性列表（File access property list）：**

文件存取特性列表通常被用来控制对文件 I/O 表现采取不同的方法。本教材使用的缺省文件存取特性列表是 H5P_DEFAULT。

详情见《HDF 用户指南》中的 The File Interface (H5F) 小节。

创建和关闭一个 HDF5 文件的步骤如下：

1. 如果需要，指定文件创建和存取特性列表。
2. 创建文件。
3. 如需要，关闭文件和关闭特性列表

要创建一个 HDF5 文件，调用程序必须包含调用创建和关闭文件。如下例：

C:

```
file_id = H5Fcreate (filename, access_mode, create_id, access_id);
status = H5Fclose (file_id);
```

FORTRAN:

```
CALL h5fcreate_f (filename, access_mode, file_id, hdferr, &
          creation_prp=create_id, access_prp=access_id)
      or
CALL h5fcreate_f (filename, access_mode, file_id, hdferr)

CALL h5fclose_f (file_id, hdferr)
```

在 FORTRAN 程序中，文件创建特性列表 creation_prp、和文件存取特性列表 access_prp，是可选参数；如果使用了缺省值，它们可以被忽略。

4.2 程序例子（Programming Example）

4.2.1 描述（Description）

下面的例子表明如何创建和关闭一个 HDF5 文件。用 C 创建一个名为 file.h5 的文件，和用 FORTRAN 创建 filef.h5 文件，然后关闭它。

[C Example] -- h5_crtfile.c

[FORTRAN Example] -- fileexample.f90

注意：要下载一个 tar 文件，包括所有程序例子和一个 Makefile，请转到 References。

4.2.2 备注（Remarks）

- 对 C： 头文件 hdf5.h 包括定义和声明，使用 HDF5 库的任何程序必须把它包括进来。对 **FORTRAN**： 模块 HDF5 包含定义和声明，任何使用 HDF5 库的程序必须用它。
- H5Fcreate/h5fcreate_f 创建一个 HDF5 文件和返回文件标号。
- *C*:
- hid_t H5Fcreate (const char *name, unsigned access_mode, hid_t creation_prp,
- hid_t access_prp)
- *FORTRAN*:
- h5fcreate_f (name, access_mode, file_id, hdferr, creation_prp, access_prp)
-
- name CHARACTER(LEN=*)
- access_flag INTEGER
- (Valid values: H5F_ACC_RDWR_F, H5F_ACC_RDONLY_F,
- H5F_ACC_TRUNC_F, H5F_ACC_EXCL_F, H5F_ACC_DEBUG_F)
- file_id INTEGER(HID_T)
- hdferr INTEGER
- (Valid values: 0 on success and -1 on failure)
- creation_prp INTEGER(HID_T), OPTIONAL
- (Default value: H5P_DEFAULT_F)
- access_prp INTEGER(HID_T), OPTIONAL
- (Default value: H5P_DEFAULT_F)
-
  - *name* 参数指定要创建的文件名。
  - 存取模式参数指明文件模式。假如文件已经存在，H5F_ACC_TRUNC (H5F_ACC_TRUNC_F in FORTRAN)将截取一个文件。
  - *creation_prp* 参数说明文件创建列表。对于 C 语言，用 H5P_DEFAULT 表示使用了缺省的文件创建列表。这个选项在 FORTRAN 中是可选的；假如没有此项，缺省的文件创建列表 H5P_DEFAULT_F，就被使用。
  - *access_prp* 参数说明文件存取特性列表。对于 C 语言，用 H5P_DEFAULT 表明缺省文件创建特性列表被使用。此项在 FORTRAN 中是可选的；如果被忽略，缺省的文件创建特性列表 H5P_DEFAULT_F，就被使用。
  - 对于 C，如果成功则返回文件标识号，否则返回一个负数。对于 FORTRAN，文件标识号是通过 *file_id* 的参数里返回的。如果调用成功，*hdferr* 返回一个 0（零）值。否则在将返回一个-1 值。
- 当一个文件不再被程序存取，H5Fclose/h5fclose_f 必须被调用来释放被这个文件使用的资源。这个调用是强制性的。
- *C*:
- herr_t H5Fclose (hid_t file_id)
-

- *FORTRAN*:
- h5fclose_f(file_id, hdferr)
- 当一个文件被创建时，根下群组（root group）就被自动创建。 每个文件都有一个根下群组，群组的路径名通常都是“/”。

4.2.3 文件内容（File Contents）

HDF 开发组已经开发出了检查 HDF5 文件内容的工具。本教材中使用的工具是 HDF5 dumper、h5dump，可以把文件内容显示成可读的形式。根据 HDF5 DDL 语法，h5dump 的输出形式是有格式的 ASCII 码。在 DDL in BNF for HDF5 中使用 Backus-Naur 格式定义了语法。

查看文件内容，键入命令：

```
h5dump <filename>
```

图 4.1 使用一个直接图表来表示文件 file.h5 (filef.h5) 的内容。本教材中的直接图表用椭圆表示 HDF5 的群组，用长方形表示数据集。箭头表示内容的包含方向。

Fig. 4.1 *file.h5 (filef.h5)* 文件的内容

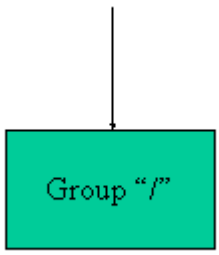


图 4.2 是文件 file.h5 的文本描述，与 h5dump 生成的一样。HDF5 文件调用了含有一个群组的 file.h5 的文件，或调用根下的群组。

Fig. 4.2 *file.h5 in DDL*

```
HDF5 "file.h5" {
GROUP "/" {
}
}
```

4.2.4 DDL 中的文件定义（File Definition in DDL）

图 4.3 是创建 HDF5 文件时简化了的 DDL 文件定义。为了简明，本教材使用了一个简化的 DDL。更完整和更严格的 DDL 在《HDF 用户指南》的 DDL in BNF for HDF5 小节里。

Fig. 4.3 *HDF5 文件定义*

DDL 中使用了下面的符号定义：

::=	定义为
<tname>	tname 名的记号
<a> | <b>	<a> 或 <b>
<a>*	<a>的 0 次或多次产生

用于文件定义而简化的 DDL 如下：

```
<file> ::= HDF5 "<file_name>" { <root_group> }

<root_group> ::= GROUP "/" { <group_attribute>* <group_member>* }

<group_attribute> ::= <attribute>

<group_member> ::= <group> | <dataset>
```

5、创建数据集（CREATING A DATASET）

5.1 什么是数据集（What is a Dataset）

数据集是数据元素的多维数组，同时有支持元数据的能力。要创建一个数据集，应用程序必须指明数据集的位置、数据集的名字、数据类型和数组的数据大小，以及数据集的创建特性列表。

5.2 数据类型（Datatypes）

数据类型是数据类型特性的集合，它能被存放在磁盘里，作为一个整体，为数据转换的目标或来源提供完整的数据类型信息。

HDF5 里有两种数据类型：元数据类型（atomic）and 复合数据类型（compound）。 元数据类型在 API 的层面上是不能分解成更小的数据类型单位它包括整型（integer），浮点型（float），日期和时间（date and time），字串（string），比特域（bitfield），和非透明（opaque）数据类型。复合数据类型是一个或多个元数据类型（和/或这些数据类型小的数组）的集合。

图 5.1 表明了 HDF5 数据类型。图 5.2a 和图 5.2b 列出了一些 HDF5 预定义的元数据类型。本教材中，只考虑了 HDF5 的预定义整型。关于更详细的数据类型，请参见《HDF5 用户指南》中 The Datatype Interface (H5T) 小节。

Fig 5.1 *HDF5 datatypes*

```
                                  +--  integer
                                  +--  floating point
                  +---- atomic  ----+--  date and time
                  |                 +--  character string
HDF5 datatypes --|                 +--  bitfield
                  |                 +--  opaque
                  |
                  +---- compound
```

Fig. 5.2a *Examples of HDF5 predefined datatypes*

Datatype	Description
H5T_STD_I32LE	Four-byte, little-endian, signed, two's complement integer
H5T_STD_U16BE	Two-byte, big-endian, unsigned integer
H5T_IEEE_F32BE	Four-byte, big-endian, IEEE floating point
H5T_IEEE_F64LE	Eight-byte, little-endian, IEEE floating point
H5T_C_S1	One-byte, null-terminated string of eight-bit characters

Fig. 5.2b *Examples of HDF5 predefined native datatypes*

Native Datatype	Corresponding C or FORTRAN Type
C:	
H5T_NATIVE_INT	int
H5T_NATIVE_FLOAT	float
H5T_NATIVE_CHAR	char
H5T_NATIVE_DOUBLE	double
H5T_NATIVE_LDOUBLE	long double
FORTRAN:	
H5T_NATIVE_INT	integer
H5T_NATIVE_REAL	real
H5T_NATIVE_DOUBLE	double precision
H5T_NATIVE_CHAR	character

5.3 数据集和数据空间（Datasets and Dataspaces）

数据空间描述了数据数组的维度。数据空间或者是数据点的有规则的 N 维数组（称为简单数据空间），或者由其它方式组成的更加通用的数据集合（称为复杂数据空间）。图 5.3 显示了数据空间。本教材中，只考虑简单数据空间。

Fig 5.3 *HDF5 dataspaces*

```
                    +-- simple
HDF5 dataspaces --|
                    +-- complex
```

数据集的维数可以是固定的（非变化），或者是可变的（可扩展）。数据空间也可以描述一个数据集的一部分，这就使得对选取的数据集进行局部 I/O 操作成为可能。

5.4 数据集创建特性列表（Dataset Creation Property Lists）

创建一个数据集时，HDF5 允许用户指定原始数据在磁盘上如何组织和/或压缩。这个信息存放在数据集创建特性列表中，并传递给数据集接口。磁盘上的这个原始数据集能被连续存放（与内存管理中的线性方式相同）、分成大块（chunks）存放、外部存放等。本教材中，缺省的数据集创建特性列表是连续的非压缩方式。更详细的关于数据集创建特性列表，参见《HDF5 用户指南》中的 The Dataset Interface (H5D) 小节。

在 HDF5 中，数据类型和数据空间是独立的对象，它们分别由任何有关的数据集创建。因为这点，数据集的创建需要数据类型和数据空间的定义。本教材中，使用了 HDF5 预定义的数据类型(integer)和仅考虑简单的数据空间。今后，只有数据空间对象的创建才是必须的。

要创建一个空的数据集（不写数据），参照如下步骤:

1. Obtain the location identifier where the dataset is to be created.
2. Define the dataset characteristics and the dataset creation property list.
   - Define a datatype.
   - Define a dataspace.
   - Specify the dataset creation property list.
3. Create the dataset.
4. Close the datatype, the dataspace, and the property list if necessary.
5. Close the dataset.

要创建一个简单的数据空间，调用程序必须调用创建和关闭此数据集。见如下例程：

C:

```
space_id = H5Screate_simple (rank, dims, maxdims);
status = H5Sclose (space_id );
```

FORTRAN:

```
CALL h5screate_simple_f (rank, dims, space_id, hdferr, maxdims=max_dims)
      or
CALL h5screate_simple_f (rank, dims, space_id, hdferr)

CALL h5sclose_f (space_id, hdferr)
```

要创建一个简单的数据集，调用程序必须调用创建和关闭此数据集。见如下例程：

C:

```
dset_id = H5Dcreate (hid_t loc_id, const char *name, hid_t type_id,
                     hid_t space_id, hid_t creation_prp);
status = H5Dclose (dset_id);
```

FORTRAN:

```
CALL h5dcreate_f (loc_id, name, type_id, space_id, dset_id, &
                  hdferr, creation_prp=creat_plist_id)
      or
CALL h5dcreate_f (loc_id, name, type_id, space_id, dset_id, hdferr)

CALL h5dclose_f (dset_id, hdferr)
```

假如在 FORTRAN 中要使用预定义的数据类型，必须调用对这个预定义的数据类型的初始化和终止访问：

```
CALL h5open_f (hdferr)
CALL h5close_f (hdferr)
```

h5open_f 必须在调用其它 HDF5 子程序库函数之前调用；h5close_f 则必须在调用其它 HDF5 子程序库函数之后调用。参见下面程序例子对这些调用的说明。

5.2 程序例子（Programming Example）

5.2.1 描述（Description）

下面例子表示怎样创建一个空的数据集。它用 C 创建了一个名为的 dset.h5(用 FORTRAN 则为 dsetf.h5)，定义了数据集的数据空间，创建了一个 4 x 6 整型数组的数据集，然后关闭数据空间、数据集和文件。

[C Example] -- h5_crtdat.c

[Fortran Example] -- dsetexample.f90

NOTE: To download a tar file of the examples, including a Makefile, please go to the References page of this tutorial.

5.2.2 备注（Remarks）

- H5Screate_simple/h5screate_simple_f 创建一个新的简单数据空间并返回一个数据空间标识号。
- *C*:
- hid_t H5Screate_simple (int rank, const hsize_t * dims,
- const hsize_t * maxdims)
- *FORTRAN*:
- h5screate_simple_f (rank, dims, space_id, hdferr, maxdims)
-
- rank INTEGER
- dims(*) INTEGER(HSIZE_T)
- space_id INTEGER(HID_T)
- hdferr INTEGER
- (Valid values: 0 on success and -1 on failure)
- maxdims(*) INTEGER(HSIZE_T), OPTIONAL
  - *rank* 参数指定秩（rank），即数据集维数的个数。
  - *dims* 参数指定数据集的大小。
  - The *maxdims* 参数指定数据集大小的上限。如果在 C 里此参数为 NULL（或在 FORTRAN 没有被指定），上限与由 dim 参数指定的维数大小相同。
  - 假如成功，在 C 中此函数返回数据空间标识号；否则返回一个负数。对于 FORTRAN，数据空间标识号（dataspace identifier）由 *space_id* 参数返回。如果调用成功则在 *hdferr* 中返回 0，否则返回-1。
- H5Dcreate/h5dcreate_f 在指定位置创建一个数据集并返回数据集标识号(dataset identifier)。
- *C*:
- hid_t H5Dcreate (hid_t loc_id, const char *name, hid_t type_id,
- hid_t space_id, hid_t creation_prp)
- *FORTRAN*:
- h5dcreate_f (loc_id, name, type_id, space_id, dset_id, &
- hdferr, creation_prp)
-
- loc_id INTEGER(HID_T)
- name CHARACTER(LEN=*)

- type_id INTEGER(HID_T)
- space_id INTEGER(HID_T)
- dset_id INTEGER(HID_T)
- hdferr INTEGER
- (Valid values: 0 on success and -1 on failure)
- creation_prp INTEGER(HID_T), OPTIONAL
  - *loc_id* 参数是位置标识号（location identifier）。
  - *name* 参数是要创建的数据集名。
  - *type_id* 参数指定的数据类型标识号（datatype identifier）。
  - *space_id* 参数是数据空间标识号（dataspace identifier）。
  - *creation_prp* 参数指定数据集创建属性列表。H5P_DEFAULT（C）和 H5P_DEFAULT_F （FORTRAN）指定缺省的数据集创建属性列表。在 FORTRAN 这个参数是可选项；如果省略了，那么缺省的数据集创建属性列表（default dataset creation property list）就被使用。
  - 如果成功，C 函数返回数据集标识号（dataset identifier）否则返回一个负数。FORTRAN 在 *dset_id* 中调用返回的数据集标识号。如成功在 *hdferr* 返回 0，否则返回-1。
- H5Dcreate/h5dcreate_f 创建一个空的数组并把数据初始化为 0。
- 当数据集不再被程序存取，When a dataset is no longer accessed by a program, H5Dclose/h5dclose_f 必须被调用来释放由此数据集占用的资源。这种调用是强制性的。
- *C*:
- hid_t H5Dclose (hid_t dset_id)
- *FORTRAN*:
- h5dclose_f (dset_id, hdferr)
-
- dset_id INTEGER(HID_T)
- hdferr INTEGER
- (Valid values: 0 on success and -1 on failure)

5.2.3 文件内容（File Contents）

文件 dset.h5 的内容(dsetf.h5 for FORTRAN) 如图 a 5.4 and 图 5.5a and 5.5b.

Figure 5.4 *Contents of dset.h5 (dsetf.h5)*

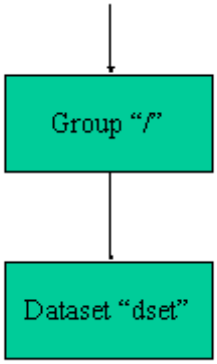


Figure 5.5a *dset.h5 in DDL*

```
HDF5 "dset.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
      DATA {
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0
      }
   }
}
}
```

Figure 5.5b *dsetf.h5 in DDL*

```
HDF5 "dsetf.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 6, 4 ) / ( 6, 4 ) }
      DATA {
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0
      }
   }
}
}
```

Note in Figures 5.5a and 5.5b that H5T_STD_I32BE, a 32-bit Big Endian integer, is an HDF atomic datatype.

5.2.4 DDL 中的数据集定义（Dataset Definition in DDL）

下面是简单的 DDL 数据集定义：

Fig. 5.6 *HDF5 Dataset Definition*

```
<dataset> ::= DATASET "<dataset_name>" { <datatype>
                                         <dataspace>
                                         <data>
                                         <dataset_attribute>* }
```

<datatype> ::= DATATYPE { <atomic_type> }

<dataspace> ::= DATASPACE { SIMPLE <current_dims> / <max_dims> }

<dataset_attribute> ::= <attribute>

6、创建一个群组（CREATING A GROUP）

6.1 什么是群组（What is a Group）？

一个 HDF5 群组是一个包含 0 个或多个 HDF5 对象的结构。HDF5 的两个基本对象是群组和数据集。要创建一个群组，调用程序必须有：

1. 获取要创建群组的位置标识号（location identifier）。
2. 创建群组。
3. 关闭群组。

要创建一个群组，程序必须调用 H5Gcreate/h5gcreate_f。要关闭这个群组，函数 H5Gclose/h5gclose_f 必须被调用。例如：

C:

```
group_id = H5Gcreate (loc_id, name, size_hint);
status = H5Gclose (group_id);
```

FORTRAN:

```
CALL h5gcreate_f (loc_id, name, group_id, error, size_hint=size)
      or
CALL h5gcreate_f (loc_id, name, group_id, error)


CALL h5gclose_f (group_id, error)
```

6.2 程序例子（Programming Example）

6.2.1 描述（Description）

下面的例子说明怎样创建和关闭一个群组。它创建一个名为 group.h5 (groupf.h5 for FORTRAN)的文件，在根下（root）创建一个名为 MyGroup 的群组，然后关闭这个群组和文件。

[C Example] - h5_crtgrp.c

[FORTRAN Example] - groupexample.f90

NOTE: To download a tar file of the examples, including a Makefile, please go to the References page.

6.2.2 备注（Remarks）

- H5Gcreate/h5gcreate_f 创建一个新的空群组，名为 MyGroup 位于根下(root)， 并返回一个群组标识号。

  C:

  ```
  hid_t H5Gcreate (hid_t loc_id, const char *name, size_t size_hint)
  ```

  FORTRAN:

  ```
  h5gcreate_f (loc_id, name, group_id, hdferr, size_hint)

          loc_id      INTEGER(HID_T)
          name        CHARACTER(LEN=*)
          group_id    INTEGER(HID_T)
          hdferr      INTEGER
                      (Possible values: 0 on success and -1 on failure)
          size_hint   INTEGER(SIZE_T), OPTIONAL
                      (Default value: OBJECT_NAMELEN_DEFAULT_F)
  ```

  - *loc_id* 参数指明要创建的群组位置。
  - *name* 参数指明要创建的群组名字。
  - *size_hint* 指定保留多少文件空间（file space）给要存储群组中将出现的名字。假如提供了一个非正值（non-positive value），则会使用缺省大小的空间（default size）。由于库函数能动态改变名字堆栈（name heap），因此可经常传递一个 0 值。

- o 对于 FORTRAN，例程的返回值在 *hdferr* 中被传递：0 表示成功，-1 表示失败。群组标识号在 *group_id* 里被传递回来。The group identifier is passed back in. 对于 C，如果成功函数就返回一个有效的群组标识号，否则返回一个负数。

- H5Gclose/h5gclose_f 关闭取组。这个调用是不可缺少的。

C:

```
herr_t H5Gclose (hid_t group_id)
```

FORTRAN:

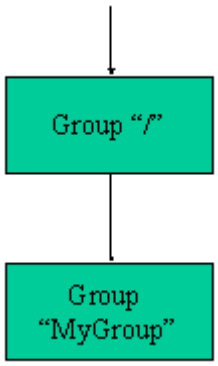
```
h5gclose_f (group_id, hdferr)

          group_id   INTEGER(HID_T)
          hdferr     INTEGER
                     (Possible values: 0 on success and -1 on failure)
```

6.2.3 文件内容（File Contents）

group.h5 的内容和群组的定义显示如下。and the definition of the group are shown below.（The FORTRAN 程序创建一个名为 groupf.h5 的 HDF 文件，在文件 groupf.h5 里的第一行显示了作为结果的 DDL。）

Fig. 8.1 *The Contents of group.h5.*		**Fig. 8.2** *group.h5 in DDL*

		HDF5 "group.h5" {
		GROUP "/" {
		GROUP "MyGroup" {
		}
		}
		}

7、创建属性（CREATING AN ATTRIBUTE）

7.1 什么是属性（Attribute）？

属性是一些小的数据集，这些数据集可用来描述与此有关对象的本性（nature）和/或用法（intended usage）。本节将介绍怎样创建、读、写一个属性。

7.1.1 创建一个属性（Creating an attribute）

创建一个属性与创建一个数据集相同。要创建一个属性，应用程序必须指定关于这个属性的对象、属性数据的数据类型和数据空间、以及属性创建特性列表（attribute creation property list）。

创建一个属性的步骤如下：

1. 获得与这个属性有关的对象标号（object identifier）
2. 定义属性的特征（characteristics）和指定属性创建特性列表。
   - 定义数据类型。
   - 定义数据空间。
   - 指定属性创建特性列表。
3. 创建属性。
4. 如果需要，关闭属性和数据类型、数据空间和属性创建特性列表。

要创建和关闭一个属性，调用程序必须使用 H5Acreate/h5acreate_f 和 H5Aclose/h5aclose_f. 例如:

C:

```
attr_id = H5Acreate (dset_id, attr_name, type_id, space_id, creation_prp);
status = H5Aclose (attr_id);
```

FORTRAN:

```
CALL h5acreate_f (dset_id, attr_nam, type_id, space_id, attr_id, &
                  hdferr, creation_prp=creat_plist_id)
```

or

```
CALL h5acreate_f (dset_id, attr_nam, type_id, space_id, attr_id, hdferr)

CALL h5aclose_f (attr_id, hdferr)
```

7.1.2 读/写属性（Reading/Writing an attribute）

属性也许只能作为一个对象来读或写；不支持局部 I/O（partial I/O）。因此，对一个属性要完成 I/O 操作，应用程序只需指定属性和属性内存的数据类型(attribute's memory datatype）。

读或写一个属性的步骤如下：

1. 获取属性标识号（attribute identifier)。
2. 指定属性内存的数据类型（attribute's memory datatype)。
3. 完成需要的操作。
4. 如需要，关闭内存数据类型（memory datatype)。

要读和/或写一个属性，调用程序必须包含 H5Aread/h5aread_f 和/或 H5Awrite/h5awrite_f 例程。见下面例子：

C:

```
status = H5Aread (attr_id, mem_type_id, buf);
status = H5Awrite (attr_id, mem_type_id, buf);
```

FORTRAN:

```
CALL h5awrite_f (attr_id, mem_type_id, buf, dims, hdferr)
CALL h5aread_f (attr_id, mem_type_id, buf, dims, hdferr)
```

7.2 编程例子（Programming Example）

7.2.1 描述（Description）

本例说明怎样创建和写一个数据集属性（dataset attribute)。打开一个已经存在的 C 文件 dset.h5（或 FORTRAN 文件 dsetf.h5)，获取 dataset /dset 标识号，定义属性的数据空间，创建数据集属性，写属性，然后关闭属性的数据空间、属性、数据集和文件。

[C Example] - h5_crtatt.c

[FORTRAN Example] - attrexample.f90

NOTE: To download a tar file of the examples, including a Makefile, please go to the References page.

7.2.2 备注（Remarks）

- H5Acreate/h5acreate_f 创建由第一个参数指定的与对象有关的属性，并返回一个标识号。

  C:

```
hid_t H5Acreate (hid_t obj_id, const char *name, hid_t type_id,
                 hid_t space_id, hid_t creation_prp)
```

  FORTRAN:

```
h5acreate_f (obj_id, name, type_id, space_id, attr_id, &
             hdferr, creation_prp)

          obj_id        INTEGER(HID_T)
          name          CHARACTER(LEN=*)
          type_id       INTEGER(HID_T)
          space_id      INTEGER(HID_T)
          attr_id       INTEGER(HID_T)
          hdferr        INTEGER
                        (Possible values: 0 on success and -1 on failure)
          creation_prp  INTEGER(HID_T), OPTIONAL
```

  - *obj_id* 参数是与属性相关的对象标识号。
  - *name* 参数是要创建的属性名。
  - *type_id* 参数是属性的数据类型标识号。
  - *space_id* 参数是属性的数据空间标识号。
  - *creation_prp* 参数是创建特性列表。对 C 有 H5P_DEFAULT（对 FORTRAN 有 H5P_DEFAULT_F）指定缺省的创建属性列表（default creation property list）。这个参数在 FORTRAN 里是可选项；当它被省略时，缺省的创建特性列表就被使用。
  - 对于 FORTRAN，如果执行成功，*hdferr* 的返回码为 0，否则为-1，属性标识号在 *attr_id* 函数里被返回。对于 C，此函数如果执行成功就返回属性标识号，否则返回一个负值。

- H5Awrite/h5awrite_f 写入完整的属性，并返回写的状态。

  C:

```
herr_t H5Awrite (hid_t attr_id, hid_t mem_type_id, void *buf)
```

FORTRAN:

```
h5awrite_f (attr_id, mem_type_id, buf, dims, hdferr)

          attr_id      INTEGER(HID_T)
          memtype_id   INTEGER(HID_T)
          buf          TYPE(VOID)
          dims         DIMENSION(*), INTEGER(HSIZE_T)
          hdferr       INTEGER
                       (Possible values: 0 on success and -1 on failure)
```

- *attr_id* 参数是要写的属性标识号。
- *mem_type_id* 参数是属性的内存数据类型（attribute's memory datatype）标识号。
- *buf* 参数是要写出的数据缓冲器（data buffer）。
- *dims* 数组保存对应的数据缓冲器维数（dimension）*buf*的大小。dim(k) 是 K 维缓冲器（k-th dimension）的值 *buf*。假如 *buf* 是一个标量（scalar），值（Values）则被忽略。
- 对于 C，如果执行成功，此函数就返回一个非负数，否则就返回一个负数。对于 FORTRAN，如果成功在 *hdferr* 参数里返回的值为 0，否则为-1。

- 当一个属性不再被程序访问（accessed），H5Aclose/h5aclose_f 必须被调用并释放其使用的属性。对于 C，如果执行成功，此函数就返回一个非负数，否则就返回一个负数。对于 FORTRAN，如果成功在 *hdferr* 参数里返回的值为 0，否则为-1。

C:

```
herr_t H5Aclose (hid_t attr_id)
```

FORTRAN:

```
h5aclose_f (attr_id, hdferr)

          attr_id   INTEGER(HID_T)
          hdferr    INTEGER
                    (Possible values: 0 on success and -1 on failure)
```

- An H5Aclose/h5aclose_f call is mandatory.

7.2.3 文件内容（File Contents）

文件 dset.h5 的内容(dsetf.h5 for FORTRAN)和属性定义如下：

Fig. 7.1a *dset.h5 in DDL*

```
HDF5 "dset.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
      DATA {
         1, 2, 3, 4, 5, 6,
         7, 8, 9, 10, 11, 12,
         13, 14, 15, 16, 17, 18,
         19, 20, 21, 22, 23, 24
      }
      ATTRIBUTE "attr" {
         DATATYPE { H5T_STD_I32BE }
         DATASPACE { SIMPLE ( 2 ) / ( 2 ) }
         DATA {
            100, 200
         }
      }
   }
}
}
```

Fig. 7.1b *dsetf.h5 in DDL*

```
HDF5 "dsetf.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 6, 4 ) / ( 6, 4 ) }
      DATA {
         1, 7, 13, 19,
         2, 8, 14, 20,
         3, 9, 15, 21,
         4, 10, 16, 22,
         5, 11, 17, 23,
         6, 12, 18, 24
```

```
        }
        ATTRIBUTE "attr" {
           DATATYPE { H5T_STD_I32BE }
           DATASPACE { SIMPLE ( 2 ) / ( 2 ) }
           DATA {
              100, 200
           }
        }
    }
}
}
```

7.2.4 DDL 中的属性定义（Attribute Definition in DDL）

Fig. 7.2 *HDF5 Attribute Definition*

```
<attribute> ::= ATTRIBUTE "<attr_name>" { <datatype>
                                          <dataspace>
                                          <data>  }
```

8、读出和写入数据集（READING FROM AND WRITING TO A DATASET）

8.1 读出和写入数据集（Reading from and Writing to a Dataset）

当处于数据集 I/O 操作时，HDF5 的库就把原始数据在内存和文件之间传递。内存中的数据可以有与文件不同的数据类型，也可以是不同的大小（即内存中的数据是数据集元素的子集，或反之亦然）。因此，要完成读或写的操作，应用程序必须指定：

- 数据集（dataset）
- 内存中数据集的数据类型（dataset's datatype in memory）
- 内存中数据集的数据空间（dataset's dataspace in memory）
- 文件里数据集的数据空间（dataset's dataspace in the file）
- 数据集传递特性列表（数据集传递特性列表控制各种各样的 I/O 操作，如参与 I/O 请求总的处理数，或把控制原始数据的高速缓存器提示给库函数。本教材里使用缺省的数据集传递特性列表）。
- 数据缓冲器（data buffer）。

读出或写入数据集的步骤如下：

1. 获取数据集标识号（dataset identifier）。
2. 指定内存数据类型（memory datatype）。
3. 指定内存数据空间（memory dataspace）。
4. 指定文件数据空间（file dataspace）。
5. 指定传递特性（transfer properties）。
6. 完成对数据集需要的操作。
7. 关闭数据集。
8. 如有必要，关闭数据空间、数据类型、和特性列表。

要读出或写入数据集，使用 H5Dread/h5dread_f 和 H5Dwrite/h5dwrite_f 例程。

C:

```
status = H5Dread (set_id, mem_type_id, mem_space_id, file_space_id,
                  xfer_prp, buf );
status = H5Dwrite (set_id, mem_type_id, mem_space_id, file_space_id,
                  xfer_prp, buf);
```

FORTRAN:

```
CALL h5dread_f(dset_id, mem_type_id, buf, dims, error, &
                  mem_space_id=mspace_id, file_space_id=fspace_id, &
                  xfer_prp=xfer_plist_id)
```

or

```
CALL h5dread_f(dset_id, mem_type_id, buf, dims,  error)


CALL h5dwrite_f(dset_id, mem_type_id, buf, dims, error, &
                  mem_space_id=mspace_id, file_space_id=fspace_id, &
                  xfer_prp=xfer_plist_id)
```

or

```
CALL h5dwrite_f(dset_id, mem_type_id, buf, dims, error)
```

8.2 编程例子（Programming Example）

8.2.1 描述（Description）

如下例子显示怎样读写一个已经存在的数据集。它打开前面例子创建的文件，获取数据集（dset）的数据集标识号，把数据集写入文件，然后读出此数据集。最后关闭此数据集和

文件。

[C Example] - h5_rdwt.c

[FORTRAN Example] - rwdsetexample.f90

NOTE: To download a tar file of the examples, including a Makefile, please go to the References page.

8.2.2 备注（Remarks）

- H5Fopen/h5fopen_f 打开一个已经存在的文件并返回一个文件标识号（file identifier）。
- *C*:
- hid_t H5Fopen (const char *name, unsigned access_mode, hid_t access_prp)
-
- *FORTRAN*:
- h5fopen_f (name, access_mode, file_id, hdferr, access_prp)
-
- name CHARACTER(LEN=*)
- access_mode INTEGER
- (Possible values: H5F_ACC_RDWR_F, H5F_ACC_RDONLY_F)
- file_id INTEGER(HID_T)
- hdferr INTEGER
- (Possible values: 0 on success and -1 on failure)
- access_prp INTEGER(HID_T), OPTIONAL
-
  - The argument *name* 是文件名。
  - The *access_mode* 参数是文件存取模式。对于 C 的 H5F_ACC_RDWR (FORTRAN 的 H5F_ACC_RDWR_F) 允许读/写存取，而 C 的 H5F_ACC_RDONLY (FORTRAN 的 H5F_ACC_RDONLY_F) 允许只读存取。
  - The *access_prp* 参数识别文件存取特性列表（file access property list）。C 的 H5P_DEFAULT 和 FORTRAN 的 H5P_DEFAULT_F 指定缺省的文件存取特性列表。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省的文件存取特性列表会被使用。
  - I 在 FORTRAN 中， 返回码是在 *hdferr* 参数里被传递回，0 表示成功，-1 表示失败。而在 C 里，如成功就返回文件标识号，否则返回一个负值。
- H5Dopen/h5dopen_f 通过 *loc_id* 指定的位置，由 *name* 指定的名字打开一个已经存在的数据集。对于 FORTRAN，返回值是在 *hdferr* 参数里被传递回，0 表示成功，-1 表示失败。而在 C 里，如成功就返回文件标识号，否则返回一个负值。

C:

```
hid_t H5Dopen (hid_t loc_id, const char *name)
```

FORTRAN:

```
h5dopen_f(loc_id, name, hdferr)

        loc_id    INTEGER(HID_T)
        name      CHARACTER(LEN=*)
        hdferr    INTEGER
                  (Possible values: 0 on success and -1 on failure)
```

- H5Dwrite/h5dwrite_f 把应用缓冲器里（application buffer）的原始数据写入到指定的数据集，并把内存中数据集（dataset in memory）的数据类型和数据空间转换成文件中数据集（dataset in the file）的数据类型和数据空间。

C:

```
herr_t H5Dwrite (hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id,
                 hid_t file_space_id, hid_t xfer_prp, const void * buf)
```

FORTRAN:

```
h5dwrite_f (dset_id, mem_type_id, buf, dims, hdferr, mem_space_id, &
                                  file_space_id, xfer_prp)

        dset_id        INTEGER(HID_T)
        mem_type_id    INTEGER(HID_T)
        buf(*,...*)    TYPE
        dims           INTEGER DIMENSION(7)
        hdferr         INTEGER
                       (Possible values: 0 on success and -1 on failure)
        mem_space_id   INTEGER(HID_T), OPTIONAL
                       (Default value: H5S_ALL_F)
        file_space_id  INTEGER(HID_T), OPTIONAL
                       (Default value: H5S_ALL_F)
        xfer_prp       INTEGER(HID_T), OPTIONAL
                       (Default value: H5P_DEFAULT_F)
```

  - *dset_id* 是数据集标识号（dataset identifier）。
  - *mem_type_id* 参数是数据集的内存数据类型标识号（dataset's memory datatype）。C 中的 H5T_NATIVE_INT（FORTRAN 中的 H5T_NATIVE_INTEGER）对库函数已经编译了的计算机是一个整型的数据类型。

- *mem_space_id* 参数是数据集的内存空间标识号（identifier of the dataset's memory dataspace）。C 中的 H5S_ALL（FORTRAN 中的 H5S_ALL_F）是缺省值并表明内存里的整个数据空间被选择为 I/O 操作。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- *file_space_id* 参数是数据集的文件数据空间标识号。C 中的 H5S_ALL（FORTRAN 中的 H5S_ALL_F）是缺省值并表明文件里数据集的整个数据空间被选择为 I/O 操作。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- *xfer_prp* 参数是数据传递特性列表标识号。C 中的 H5P_DEFAULT（FORTRAN 中的 H5P_DEFAULT_F）是缺省值并表明使用了缺省数据传递特性列表。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- *buf* 参数是要写的数据缓存器（data buffer）。
- *dims* 参数是支持对应数据缓存器维数大小的数组。
- 对于 FORTRAN， the *hdferr* 参数是传递回的错误返回码；0 表示成功，-1 表示失败。对于 C，如成功就返回文件标识号，否则返回一个负值。

- H5Dread/h5dread_f 把原始数据从指定数据集读到一个应用缓存器里(application buffer)，并把文件数据类型和数据空间（file datatype and dataspace）转换为内存数据类型和数据空间（memory datatype and dataspace）。

C:

```
herr_t H5Dread (hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id,
                hid_t file_space_id, hid_t xfer_prp, void * buf)
```

FORTRAN:

```
h5dread_f (dset_id, mem_type_id, buf, dims, hdferr, mem_space_id, &
                file_space_id, xfer_prp)

        dset_id         INTEGER(HID_T)
        mem_type_id     INTEGER(HID_T)
        buf(*,...*)     TYPE
        dims            INTEGER, DIMENSION(7)
        hdferr          INTEGER
                        (Possible values: 0 on success and -1 on failure)
        mem_space_id    INTEGER(HID_T), OPTIONAL
                        (Default value: H5S_ALL_F)
        file_space_id   INTEGER(HID_T), OPTIONAL
                        (Default value: H5S_ALL_F)
        xfer_prp        INTEGER(HID_T), OPTIONAL
                        (Default value: H5P_DEFAULT_F)
```

- The *dset_id* 参数是数据集标识号。
- The *mem_type_id* 参数使数据集的内存数据类型标识号。C 中的 H5T_NATIVE_INT (FORTRAN 中的 H5T_NATIVE_INTEGER) 对库函数已经编译了的计算机是一个整型的数据类型。
- The *mem_space_id* 参数是数据集的内存空间标识号（identifier of the dataset's memory dataspace)。C 中的 H5S_ALL（FORTRAN 中的 H5S_ALL_F）是缺省值并表明内存里的整个数据空间被选择为 I/O 操作。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- The *file_space_id* 参数是数据集的文件数据空间标识号。C 中的 H5S_ALL（FORTRAN 中的 H5S_ALL_F）是缺省值并表明文件里数据集的整个数据空间被选择为 I/O 操作。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- The *xfer_prp*参数是数据传递特性列表标识号。C 中的 H5P_DEFAULT(FORTRAN 中的 H5P_DEFAULT_F）是缺省值并表明使用了缺省数据传递特性列表。在 FORTRAN 中此参数是可选项；如果忽略了此项，缺省值会被使用。
- The *buf* 参数是要读入的数据缓存器（data buffer)。
- The *dims* 参数是支持对应数据缓存器维数大小的数组。
- 对于 FORTRAN， the *hdferr* 参数是传递回的错误返回码；0 表示成功，-1 表示失败。对于 C，如成功就返回文件标识号，否则返回一个负值。

8.2.3 文件内容（File Contents）

Figure 6.1a 是文件 dset.h5 的内容(由 C 程序创建)。
Figure 6.1b 是文件 dsetf.h5 的内容(由 FORTRAN 程序创建)。

Fig. 6.1a *dset.h5 in DDL*

```
HDF5 "dset.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
      DATA {
         1, 2, 3, 4, 5, 6,
         7, 8, 9, 10, 11, 12,
         13, 14, 15, 16, 17, 18,
         19, 20, 21, 22, 23, 24
      }
   }
```

```
      }
      }
```

Fig. 6.1b *dsetf.h5 in DDL*

```
HDF5 "dsetf.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 6, 4 ) / ( 6, 4 ) }
      DATA {
         1, 7, 13, 19,
         2, 8, 14, 20,
         3, 9, 15, 21,
         4, 10, 16, 22,
         5, 11, 17, 23,
         6, 12, 18, 24
      }
   }
}
}
```

9、复合数据类型（COMPOUND DATATYPES）

9.1 创建复合数据类型（Creating Compound Datatypes）

复合数据类型与 C 语言里的结构（struct）或 FORTRAN 语言里的公用块（common block）类似。它是一个或多个数据类型的集合，并能包括复合数据类型本身。要创建和使用复合数据类型，必须熟悉复合数据类型的各种特性：

- 属于复合类（class compound）。
- 总大小是固定的（fixed total size），单位为 bytes。
- 由 0 个或多个独立名字的成员(members)组成，在数据中占据非重叠区域(occupying non-overlapping regions)。
- 每个成员有它自己的数据类型。
- 每个成员由值为 0 到 N-1 之间的索引号（index number）引用，这里的 N 是复合数据类型里的成员数。
- 复合数据类型里每个成员都有一个名字，同类（sibling）之间是唯一的。
- 每个成员有固定字节的偏移量（fixed byte offset），位于复合数据类型里此成员

首字节（the first byte）或称最小字节地址。

- 每个成员可以是一个最高到 4 维（up to four dimensions）的小数组。

当成员被添加到复合数据类型里并且此后不能被改变时，这个复合数据类型的成员特性就被定义了。

复合数据类型必须在其它数据类型以外生成。首先创建一个空的复合数据类型并指定它的大小（total size），然后按任意顺序把这个成员添加到此复合数据类型里。

9.2 程序例子（Programming Example）

9.2.1 描述（Description）

下面例子说明如何创建一个复合数据类型，把一个数组写入使用复合数据类型的文件里，然后读回（read back）这些成员的子集（subsets of the members）。

[C Example] - h5_compound.c

[Fortran 90 Example] - compound.f90

程序的输出结果如下：

```
Field c :
1.0000 0.5000 0.3333 0.2500 0.2000 0.1667 0.1429 0.1250 0.1111 0.1000

Field a :
0 1 2 3 4 5 6 7 8 9

Field b :
0.0000 1.0000 4.0000 9.0000 16.0000 25.0000 36.0000 49.0000 64.0000 81.0000

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

9.2.2 备注（Remarks）

- H5Tcreate 创建一个含有确定字节数指定的新数据类型。
- hid_t H5Tcreate (H5T_class_t class, size_t size)
  - The *class* 参数指定要创建的数据类型。目前仅有 H5T_COMPOUND 数据类型类支持此函数。
  - The *size* 参数指明要创建的数据类型的字节数。
- H5Tinsert 添加成员到由 *type_id* 指定的复合数据类型里。

- herr_t H5Tinsert (hid_t type_id, const char * name, off_t offset,
- hid_t field_id)
  - The *type_id* 参数是要修改的复合数据类型的标识号。
  - The *name* 参数是要插入的域名（name of the field to insert）。在复合数据类型里新的成员名是唯一的。
  - The *offset* 参数是要插入的域内存结构（memory structure of the field）的偏移量。库函数定义 HOFFSET 宏（macro）来计算结构中（struct）成员的偏移量：
  - HOFFSET (s, m)

    在结构 s 中，宏（macro）计算成员 m 的偏移量。

  - The *field_id* 是要插入的域（field）数据类型标识号。
- H5Tclose 释放数据类型。
- herr_t H5Tclose (hid_t type_id)

The *type_id* 参数是要释放的数据类型标识号。

9.2.3 文件内容（File Contents）

```
HDF5 "SDScompound.h5" {
GROUP "/" {
   DATASET "ArrayOfStructures" {
      DATATYPE {
         H5T_STD_I32BE "a_name";
         H5T_IEEE_F32BE "b_name";
         H5T_IEEE_F64BE "c_name";
      }
      DATASPACE { SIMPLE ( 10 ) / ( 10 ) }
      DATA {
         {
            [ 0 ],
            [ 0 ],
            [ 1 ]
         },
         {
            [ 1 ],
            [ 1 ],
            [ 0.5 ]
         },
         {
```

```
            [ 2 ],
            [ 4 ],
            [ 0.333333 ]
        },
        {
            [ 3 ],
            [ 9 ],
            [ 0.25 ]
        },
        {
            [ 4 ],
            [ 16 ],
            [ 0.2 ]
        },
        {
            [ 5 ],
            [ 25 ],
            [ 0.166667 ]
        },
        {
            [ 6 ],
            [ 36 ],
            [ 0.142857 ]
        },
        {
            [ 7 ],
            [ 49 ],
            [ 0.125 ]
        },
        {
            [ 8 ],
            [ 64 ],
            [ 0.111111 ]
        },
        {
            [ 9 ],
            [ 81 ],
            [ 0.1 ]
        }
    }
}
```

NSMC

```
}
}
```

NSMC