Options

-s	single step through the images. Once an image is displayed, it remains on the screen until you press **RETURN** to display the next image or enter **Q** to quit.
-l	makes the image as large as possible on the screen.
-p *xloc yloc*	places the upper left corner of the display at position (`xloc`, `yloc`) on the screen.
-e *expansion*	expands the image by the expansion factor specified by `expansion`.

The syntax for `hdfrseq` is exactly the same.

Example 6

Five images are stored in RIS8 form in the files `star1`, `star2`, `star3`, `star4`, and `star5`. You want to use `hdfrseq` to display all five images, in sequence, but with a pause between them. To make them appear on screen three times their normal size, enter:

```
hdfrseq -s -e 3 star1 star2 star3 star4 star5
```

hdftopal: Extracting a palette from an HDF file

hdftopal converts a palette from an HDF file to a raw palette in a raw palette file. The outgoing palette will have 768 bytes: first 256 red values, then 256 greens, then 256 blues.

The syntax for hdftopal is:

```
hdftopal hdffile rawpalfile
```

hdffile, the input file, is an HDF file containing a palette.

rawpalfile is a 768-byte output file with the red, green, and blue components stored as described above.

hdfrseq and hdfseq: Displaying Images

The utilities *hdfseq* and *hdfrseq* display sequences of images, one after the other, from files containing raster image sets.

Although hdfseq and hdfrseq perform essentially the same function, the situations in which they are used are different. hdfseq displays images on a Sun-3 console and on any SGI IRIS that has gllib, when those images are stored at the Sun 3 or IRIS workstation. When hdfseq executes on a file, it causes the raster image to be displayed immediately on the console.

hdfrseq performs the same operation, only remotely, via a terminal emulator such as Telnet 2.3 or Teltool 1.2. (The "r" in hdfrseq stands for *remote*.) When hdfrseq executes on a file, it converts the image from RIS8 format to NCSA's ICR (interactive color raster) format and sends it to the receiving terminal. ICR is a protocol that is easy to transmit to a remote display station. If the receiving software on the display station can decipher the ICR protocol, it displays the image on the console.

In a typical application, the display station would be running a program, such as NCSA Telnet (Version 2.3 or later) or NCSA Teltool (Version 1.2 or later), that understands ICR and immediately converts the incoming stream to a screen display.

The syntax for hdfseq is:

```
hdfseq [-s] [-l] [-p xloc yloc] [-e expansion] file1 file2 ...
```

file1, file2, and so forth are HDF files that contain raster image sets. Only one image is displayed per file. If more than one file is listed, the corresponding images are displayed in sequence, that is, in the order in which the files are listed. Each new image overwrites the preceding image on the screen.

A file called `denm.hdf` contains three 512 x 256 images and three palettes. To extract these images and palettes and put them in separate files, enter:

```
hdftor8 denm.hdf
```

Six files are produced with the names img1.512.256, img2.512.256, img3.512.256, pal.1, pal.2, and pal.3.

r24hdf8: Converting 24-bit Raster Image to HDF 8-Bit Raster Images

The command line utility `r24hdf8` quantizes a raw RGB 24-bit pixel image into an 8-bit image with a 256-color palette and stores both the palette and image in an HDF file.

The syntax for `r24hdf8` is:

```
r24hdf8 x_dim y_dim r24_file hdf8_file
```

`x_dim` is the x dimension of the image (horizontal zize).

`y_dim` is the y dimension of the image (vertical zize).

`r24_file` is a file containing the raw RGB 24 bit image. The order of pixels is left to right and top to bottom. Each pixel is three contiguous bytes: red byte, green byte, and blue byte. Use filter `ptox` to convert from "planar" to "pixel" where *planar* means all red bytes for the whole image, followed by all green bytes for the whole image, followed by all blue bytes for the whole image.

`hdf8_file`, the output file, is an HDF file with one 8-bit raster image set; i.e. 8-bit image, dimensions, and RGB palette.

paltohdf: Converting a Raw Palette to HDF

`paltohdf` converts a raw palette to HDF format. The incoming palette is assumed to have 768 bytes organized in the following order: 256 red values, 256 green values, and 256 blue values. The palette in the HDF file will have the RGB values interlaced: RGB RGB RGB... This is standard HDF format for 8-bit palettes.

The syntax for `paltohdf` is:

```
paltohdf rawpalfile hdffile
```

`rawpalfile` is a 768-byte input file with the red, green, and blue components stored as described above.

`hdffile`, the output file, is an HDF file. If *`hdffile`* does not exist, it is created and the palette is added to it. If it already exists, the palette is appended to it.

them in `pic.hdf`, enter:

```
r8tohdf 300 400 pic.hdf -i pic1 pic2 pic3
```

Example 4

A combination of different types of raster image sets is to be stored in a file called `ras.hdf`. The image from file `rawras1` is to be stored without a palette. The images from `rawras2` are to be stored with a palette that is copied from a file called `mypal`. The images from `rawras1` and `rawras2` are to be compressed using run length encoding, and `rawras3` is not to be compressed. All images are 256 x 512.

```
r8tohdf 256 512 ras.hdf -c rawras1 -p mypal rawras2 -r rawras3
```

hdftor8: Extracting 8-Bit Raster Images and Palettes from an HDF File

The utility *hdftor8* extracts the images and or palettes from an HDF file and stores them in two sets of files that contain only images and palettes, respectively. The syntax for `hdftor8` is as follows:

```
hdftor8 hdf_file [-i] [-v] [-r image_file][-p palette_file]
```

Where *hdf_file* is the file to extract images and palettes from, and *image_file* and *palette_file* are basic names of the files that will contain the images and palettes. These names are extended as follows: For each image file, the filename is given the extension ".#.@.%", where # stands for the image number from the original file, @ is the x dimension of the image, and % is the y dimension of the image. For each palette file, the filename is given the extension ".#", where # stands for the palette number from the original file.

If no names are given for the image file, the default name "img.#.@.%" is used, where #, @ and % are defined as in the preceding paragraph. Similarly the default name for a palette file is "pal.#".

Options

-i puts the program in interactive mode, so you can specify filenames interactively.

-v specifies verbose mode, providing descriptive messages.

-r indicates that the file whose name follows is to hold images.

-**p** indicates that the file whose name follows is to hold palettes.

Example 5

Utilities for Working with Raster Image Sets

There are three utility programs for working with HDF files that contain raster image sets. These routines can be executed interactively at the command level without being embedded in programs.

r8tohdf: Converting 8-Bit Raster Images to HDF

The utility `r8tohdf` converts one or more raw images to HDF RIS8 format and writes them to a file. The syntax for storing one RIS8 in a file using `r8tohdf` is as follows:

```
r8tohdf rows cols outfile [-p palfile] {[-c],[-r],
[-i]} imf1 imf2 ...
```

where

- *rows* and *cols* are the number of rows and columns, respectively, of the raster image
- *outfile* is the file that will contain the raster image set, and
- *imf1, imf2,* and so forth, are files containing 8-bit raw raster images.

Options

-p palfile optionally stores a file containing a palette in the RIS8. If -p is not specified, no palette is stored in the RIS8.

{[-c],[-r],[-i]} optionally chooses compression by run length encoding (-c), compression by the IMCOMP method (-i), or no compression (-r). The default is -r.

Example 1

A 256 x 512 byte raw raster image is contained in a file called `rawras`, and the palette with which it is to be used is stored in a file called `mypal`. To convert this information to an RIS8 without compression and store the result in a file called `ras.hdf`, enter:

```
r8tohdf 256 512 ras.hdf -p mypal rawras
```

Example 2

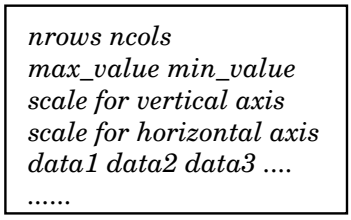
A 800 x 1000 byte raw raster image is stored in a file called `bigpic`. You want to first convert this information to an RIS8 with run length encoding for compression and no palette, and then store the result in a file called `bigpic.hdf`. Enter:

```
r8tohdf 800 1000 bigpic.hdf -c bigpic
```

Example 3

Three 300 x 400 raw images are contained in files `pic1`, `pic2`, and `pic3`. To convert all three files to RIS8s with imcomp compression and store

Figure 7.3 Format Used in a Text File for Input



NOTE: There are *nrows* vertical axis scale values and *ncols* horizontal axis scale values. *data1*, *data2*, etc., are the floating-point data and are assumed to be ordered by rows, left to right and top to bottom.

Examples

Example 1

Convert floating-point data in file f1.txt to a SDS format, and store it as an SDS in HDF file o1:

```
fptohdf f1.txt –o o1
```

Example 2

Convert floating-point data in file f2 to an 8-bit raster image and store it as an RIS8 in file o2:

```
fptohdf f2 –o o2 –r
```

Example 3

Convert floating-point data in file f3 to RIS8 format and SDS format and store both the RIS8 and the SDS in file o3:

```
fptoHdf f3 –o o3 –r –f
```

Example 4

Convert floating-point data in file f4 to a 500 x 600 raster image, storing the corresponding RIS8 in file o4. Also store a palette from `palfile` with the image:

```
fptohdf f4 –o o4 –r –e 500 600 –p palfile
```

Example 5

Convert floating-point data in all files whose names begin with the letter ‘f‘ to 500 x 600 images and store the corresponding RIS8s in file o5:

```
fptohdf f* –o o5 –r –i 500 600
```

Options

-r Stores as image in raster image set in output file

-e Expands float data via pixel replication to produce image (default if -I option chosen)

-i Applies bilinear interpolation when expanding float data to produce image

<horiz> <vert>

when **-e** or **-i** options are chosen, these give the resolution in pixels of the desired image

-p *palfile*
Stores palette with image; get palette from HDF file *palfile.*

-m <mean>
Causes the data to be scaled around <mean> when generating the image, according to the following formulas:

```
newmax = mean + 0.5*max(abs(max-mean),abs(mean-min))
newmin = mean - 0.5*max(abs(max-mean),abs(mean-min))
```

-f Stores as scientific dataset in output file (default)

Notes

- Image expansion is available via either pixel replication (-e) or bilinear interpolation (-i), but not both.
- If the "-i" option is chosen, the expanded image must have dimensions that are greater than or equal to the dimensions of the original dataset
- An optional palette can accompany the image by loading it from an HDF file that contains a palette.
- Data from several input files (one set per input file) are stored as several datasets and/or images in one output file. Alternatively, a shell script can be used to call `fptohdf` repeatedly to convert data from several input files to several corresponding HDF files.
- If an HDF file is used for input, it must contain an SDS. The SDS need only contain a dimension record and the data, but if it also contains `max` and `min` values and/or scales for the horizontal and vertical axes, these will be used also.

  ("Scales" are used for determining the gaps between the points on the axes. If the gaps are all the same, a uniform scale is given, such as "1.0 2.0 3.0 ... n". In an HDF file, scales may be omitted, but in a text file (see below) they must be included.)
- If a text file is used for input, it must follow the format shown in Figure 7.3.

```
#!/bin/csh -f
set file=$1
shift
hdfed -batch $file -nobackup << EOF
info -all -group $*
close
quit
EOF
echo ""
```

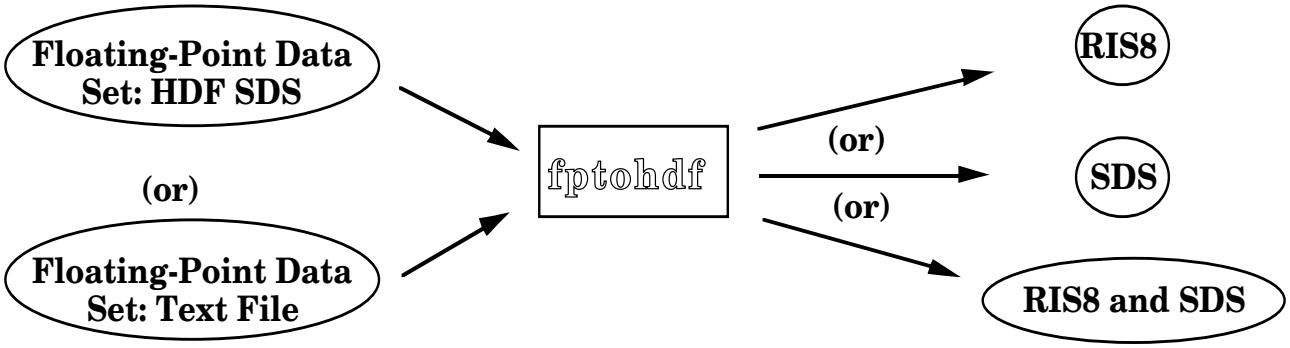
This shell script lists information about the groups in an HDF file. The "-batch" in the line that calls `hdfed` tells `hdfed` that the edit commands are to be taken from the stream of characters beginning on the next line and ending with the EOF symbol.

fptohdf: Converting Floating-Point Data to SDS and/or RIS8

Basics

fptohdf is a utility that converts floating-point arrays (from either text files or HDF scientific datasets) to either HDF 8-bit raster image sets (RIS8) or HDF scientific datasets (SDS), or both, and stores the results in an HDF file (see Fig. 7.2). The image can be scaled about a mean value that is provided by the user.

Figure 7.2 The fptohdf Utility



The syntax of `fptohdf` is as follows:

```
fptohdf infile [infile ...] -o outfile [out_opts]

out_opts:
   [[-r] [[-e|-i] <horiz> <vert>] [-p palfile]
   -m <mean>] [-f]
```

- *infile* is a file containing a single floating-point array in either text or HDF SDS format. If the format is text, see "Notes" below on how it must be organized.

- *outfile* is a file containing the converted dataset in HDF format. Depending on *out_opts*, *outfile* contains a scientific dataset and/or raster image set for each of the datasets in the input files.

```
he> ! groups we can use the select command
he>
he> select -help
select [<predicates>]
<commands>*
end
Steps through all elements in the file that satisfies the
predicates and execute the commands on them.
he>
he> select tag=306
>> putr8 -image testImages# -palette testPalettes# -verbose
>> end
Writing to file: testImages5
Writing to file: testPalettes5
Writing to file: testImages12
Writing to file: testPalettes12
Writing to file: testImages19
Writing to file: testPalettes19
he>
he> ! select and if commands take the same predicates as
he> ! next and pref.  There are also the predicates
he> ! "succeed" and "fail" that test the return status
he> ! of the last command.
he>
```

put

```
he> ! put puts an element into a binary file
he> ! This is a dumb routine and does not know about the
he> ! formats of the element
he>
he> put -help
put [-file <file>] [-verbose]
Put the raw binary of this element in a file
-file          Out file name (default "element#.@")
-verbose       Output diagnostic info
he>
he> put -file binary#
he> put -file myBinary -verbose
Writing to file: myBinary
he>
he> ! that's it for now
he> revert;close;quit
zaphod|2%
```

---

hdfed with the -batch Option

Sometimes we want to have `hdfed` execute a series of commands, without waiting for a prompt between commands. This process is useful when a certain sequence of `hdfed` commands is commonly used, as illustrated in the following shell script:

```
he> ! Let's look at the file 'test'.
he> close;open test;info -all
(1) Scientific Data     (SciData)   : (tag 702), Ref: 1
(2) Scientific Data     (SciData)   : (tag 702), Ref: 2
(3) Scientific Data     (SciData)   : (tag 702), Ref: 3
(4) Number type         (Utility)   : (tag 106), Ref: 3
(5) SciData description (SciData)   : (tag 701), Ref: 3
(6) SciData scales      (SciData)   : (tag 703), Ref: 3
(7) SciData max/min     (SciData)   : (tag 707), Ref: 3
*(8) Scientific Data Group (SciData) : (tag 700), Ref: 3
Empty (tag 1) : 8 slots.
he>
he> close;
he>
```

display

```
he> !We will open a file with some RIS8 images.
he>
he> open denm.HDF
he> display
he>
he> ! display displays the current r8 group image via ICR.
he> ! I.e. if you are using NCSA Telnet on a MacII, this
he> ! would display the images from denm.hdf on your screen
he> ! NOTE: not guaranteed to work otherwise.
he>
```

putr8

```
he> ! putr8 puts an r8 group into raw files
he> putr8 -help
putr8 [-image <image>] [-palette <pal>] [-verbose]
Put an r8 group into raw image and palette files
-image         Image file name template (default "img#.@.%")
-palette       Palette file name template (default "pal#")
-verbose       To give output of steps taken
he>
he> putr8 -image my_image#.@.% -palette my_palette -verbose
Writing to file: my_image5.512.256
Writing to file: my_palette
he>
```

getr8

```
he> ! getr8 works in the reverse fashion
he> getr8 -help
getr8 <image> <xdim> <ydim> [-palette <palette>] [-raster|-rle|- imcomp]
Get a r8 group from raw files
-palette       Raw palette file
-rasterNo compression (default)
```

Figure 7.1 Tutorial Session (Continued)

```
-rle           Run-length compression
-imcomp Imcomp compression
he>
```

select

```
he> ! To step through a file and, say, putr8 on all r8
```

```
-editorUse editor (default EDITOR env value)
he>
he> annotate -editor /usr/ucb/ex
"/tmp/he965.0" 1 line, 32 characters
:p
testing 1 2 3 4 5 6 7 8 9 0....
:s/$/<some additional stuff>/
testing 1 2 3 4 5 6 7 8 9 0....<some additional stuff>
:wq
"/tmp/he965.0" 1 line, 55 characters
he> info -label
(7) Scientific Data Group (SciData) : (tag 700), Ref: 1
Label: testing 1 2 3 4 5 6 7 8 9 0....<some additional stuff>
he>
he> annotate -descriptor -editor /usr/ucb/ex
"/tmp/he965.1" 1 line, 48 characters
:p
gdshjfhjdf asdgy sdf sdgf sdfg sgdfh as dbf asl
:c
This is the descriptor, it can have non-graphic characters
.
:wq
"/tmp/he965.1" 1 line, 58 characters
he>
he> info -all
(1) Machine type        (Utility)   : (tag 107), Ref: 4369
...
*(7) Scientific Data Group (SciData) : (tag 700), Ref: 1
```

Figure 7.1 Tutorial Session (Continued)

```
(8) Data Id Label       (Utility)   : (tag 104), Ref: 2
(9) Data Id Label       (Utility)   : (tag 104), Ref: 3
(10) Data Id Annotation  (Utility)   : (tag 105), Ref: 4
(11) Data Id Annotation  (Utility)   : (tag 105), Ref: 5
(12) Data Id Annotation  (Utility)   : (tag 105), Ref: 6
(13) Data Id Annotation  (Utility)   : (tag 105), Ref: 7
Empty (tag 1) : 3 slots.
he>
```

write

```
he> ! Write to another HDF file
he>
he> write -help
write <file> [-attachto <atag> <aref>]
Write an element or group into another HDF file
  -attachto:     ONLY for writing annotations
  <atag>/<aref>: What element to attach annotation to
he>
he> write test
he>
```

```
Number type         (Utility)   : (tag 106), Ref: 1
Data Id Label       (Utility)   : (tag 104), Ref: 2
Data Id Label       (Utility)   : (tag 104), Ref: 3
Data Id Annotation  (Utility)   : (tag 105), Ref: 4
Empty (tag 1) : 6 slots.
he> delete
he> ! This deletes the Scientific Data Group.
he> info -all
(1) Machine type        (Utility)   : (tag 107), Ref: 4369
(3) Number type         (Utility)   : (tag 106), Ref: 1
(8) Data Id Label       (Utility)   : (tag 104), Ref: 2
(9) Data Id Label       (Utility)   : (tag 104), Ref: 3
```

Figure 7.1 Tutorial Session (Continued)

```
(10) Data Id Annotation  (Utility)   : (tag 105), Ref: 4
Empty (tag 1) : 11 slots.
he>
he> ! Let's delete an element.
he>
he> next tag=104 ref=3
he> info -all
(1) Machine type        (Utility)   : (tag 107), Ref: 4369
(3) Number type         (Utility)   : (tag 106), Ref: 1
(8) Data Id Label       (Utility)   : (tag 104), Ref: 2
*(9) Data Id Label       (Utility)   : (tag 104), Ref: 3
(10) Data Id Annotation  (Utility)   : (tag 105), Ref: 4
Empty (tag 1) : 11 slots.
he> delete
he> info -all
(1) Machine type        (Utility)   : (tag 107), Ref: 4369
(3) Number type         (Utility)   : (tag 106), Ref: 1
(8) Data Id Label       (Utility)   : (tag 104), Ref: 2
(10) Data Id Annotation  (Utility)   : (tag 105), Ref: 4
Empty (tag 1) : 12 slots.
he>
```

revert

```
he> ! Reverting to original, unaltered file. This
he> ! cannot work if nobackup is used when opening.
he> ! the file.
he> ! Here we revert back to the original file:
he> revert
he>
```

annotate

```
he> ! Annotations are labels and descriptors.
he>
he> annotate -help
annotate [-label|-descriptor] [-editor <editor>]
Edit an annotation
-label        Edit label (default)
-descriptor   Edit descriptor
```

```
he> ! to go back to groups
he> next group
he> info
(7) Scientific Data Group (SciData) : (tag 700), Ref: 1
he>
```

dump

```
he> ! to see the binary representation of this element
he>
he> dump -help
dump [-offset <offset>] [-length <len>]
[-decimal|-octal|-hexadecimal|-float|-ascii]
Od the present element
-offset Start offset
-length Length to look at
-decimal      Decimal format
-octal        Octal format
-hexadecimal  Hexadecimal format
-float        Float format
-ascii        Ascii format
he>
he> dump
0000000  001276 000001 001275 000001 001277 000001 001303 000001
0000020
he>
he> dump -decimal
0000000  00702 00001 00701 00001 00703 00001 00707 00001
0000020
he>
```

delete

```
he> ! deleting groups
he>
he> delete -help
delete
Delete this element or group.
he>
he> ! If an element is required by other groups it is
he> ! alone.  However, this is not perfect since the way
he> ! group membership is determined can be pretty ad hoc
he>
he> info -all -group
**Group 1:
Scientific Data Group (SciData) : (tag 700), Ref: 1
Scientific Data     (SciData)   : (tag 702), Ref: 1
SciData description (SciData)   : (tag 701), Ref: 1
SciData scales      (SciData)   : (tag 703), Ref: 1
SciData max/min     (SciData)   : (tag 707), Ref: 1

**These do not belong to any group:
Machine type        (Utility)   : (tag 107), Ref: 4369
```

```
Empty (tag 1) : 6 slots.
he>
he> ! Move in the file using next and prev.
he> ! The move direction depends on the relative positions,
he> ! so it is often necessary to do an 'info -all' first
he>
he> info -all
(1) Machine type        (Utility)   : (tag 107),          Ref: 4369
(2) Scientific Data     (SciData)   : (tag 702),          Ref: 1
(3) Number type         (Utility)   : (tag 106),          Ref: 1

(4) SciData description (SciData)   : (tag 701),          Ref: 1
(5) SciData scales      (SciData)   : (tag 703),          Ref: 1
(6) SciData max/min     (SciData)   : (tag 707),          Ref: 1
*(7) Scientific Data Group (SciData) : (tag 700), Ref: 1
(8) Data Id Label       (Utility)   : (tag 104),          Ref: 2
(9) Data Id Label       (Utility)   : (tag 104),          Ref: 3
(10) Data Id Annotation  (Utility)   : (tag 105), Ref: 4
Empty (tag 1) : 6 slots.
he>
he> ! The '*' in the first column marks the current
he> ! position.
he> ! The next and prev commands work on predicates.
he> ! Default predicate is "group".
he> ! This means that each move will move to the next/prev
he> ! group if I now want to move to the max/min element,
he> ! I can use the 'tag=' predicate
he>
he> prev tag=707
he> info
(6) SciData max/min     (SciData)   : (tag 707), Ref: 1
he>
he> ! Other predicates are ref", "any", with comparators
he> ! "=", "!=", ">", "<", ">=", "<="
he> ! For example, "group ref >= 10" is legal and matches
he> ! a group with ref>=10.
he>
he> ! This predicate persist for next and prev.
he> ! That means that if I now do another 'next'
he> ! it will look for a tag=707
he>
he> next
Reached end of file. Not moved.
```

next
prev
predicates
*

Figure 7.1 Tutorial Session (Continued)

```
he> info
(6) SciData max/min     (SciData)   : (tag 707), Ref: 1
he>
```

```
he>
he> info -help
info [-all] [-long] [-group] [-label]
-all          Show info for all elements in file
-long         Show more info
-group        Organize info in group(s)
-label        Show label if any
he>
he> info -all -label -long
(1) Machine type        (Utility)   : (tag 107)
       Ref: 4369, Offset: 0, Length: 0 (bytes)
       Label: Sun Machine type??
(2) Scientific Data     (SciData)   : (tag 702)
       Ref: 1, Offset: 202, Length: 40000 (bytes)
(3) Number type         (Utility)   : (tag 106)
       Ref: 1, Offset: 40202, Length: 4 (bytes)
(4) SciData description (SciData)   : (tag 701)
       Ref: 1, Offset: 40206, Length: 22 (bytes)
(5) SciData scales      (SciData)   : (tag 703)
       Ref: 1, Offset: 40228, Length: 802 (bytes)
(6) SciData max/min     (SciData)   : (tag 707)
       Ref: 1, Offset: 41030, Length: 8 (bytes)
*(7) Scientific Data Group (SciData) : (tag 700)
       Ref: 1, Offset: 41038, Length: 16 (bytes)
       Label: testing 1 2 3 4 5 6 7 8 9 0....<additional stuff>
(8) Data Id Label       (Utility)   : (tag 104)
       Ref: 2, Offset: 41275, Length: 53 (bytes)
(9) Data Id Label       (Utility)   : (tag 104)
```

Figure 7.1 Tutorial Session (Continued)

```
       Ref: 3, Offset: 41230, Length: 22 (bytes)
(10) Data Id Annotation  (Utility)   : (tag 105)
       Ref: 4, Offset: 41252, Length: 23 (bytes)
Empty (tag 1) : 6 slots.
he> info -group -all
**Group 1:
Scientific Data Group (SciData) : (tag 700), Ref: 1
Scientific Data     (SciData)   : (tag 702), Ref: 1
SciData description (SciData)   : (tag 701), Ref: 1
SciData scales      (SciData)   : (tag 703), Ref: 1
SciData max/min     (SciData)   : (tag 707), Ref: 1

**These do not belong to any group:
Machine type        (Utility)   : (tag 107), Ref: 4369
Number type         (Utility)   : (tag 106), Ref: 1
Data Id Label       (Utility)   : (tag 104), Ref: 2
Data Id Label       (Utility)   : (tag 104), Ref: 3
Data Id Annotation  (Utility)   : (tag 105), Ref: 4
```

annotate	Annotate an object
if	Conditional statement
select	Loop for each object
alias	Set or show aliases
unalias	Remove an alias
wait	Message and wait for return

Tutorial Session

In the absence of a user's manual for `hdfed`, this document contains a tutorial in the form of a session on the editor. Examples of almost all of the commands are given below. In the following script, bold characters represent those typed by a user. Plain text characters represent characters that were printed by the computer.

Figure 7.1 Tutorial Session

```
zaphod|2% he
he> ! This is a script of a session on the HDF editor.
he> ! It is meant as an example for anyone learning to use
he> ! the editor.
he> ! The exclamation mark in the FIRST column starts
!   he> ! a comment
he>
help he> ! help command
he> help
Type <command> -help for usage of command
e.g. "open -help" give help on the open command
DO NOT just type the command and expect a help line
Some commands like delete do not need any argument
If you are just starting to learn, try this program on
a file that is expendable

List of commands:
________
open          Open HDF file
close         Close HDF file
...
...
open he> ! opening a file
he>
he> open -help
open <file> [-nobackup]
-nobackup     Don't make a backup for this file.
he>

he> open h1
he>
info he> ! look into the contents of this file
he> ! using info
```

- **-batch**
  Specifies that input to `hdfed` is to be input via a stream of `hdfed` commands, rather than interactively.

To receive usage information, as well as a quick list of the `hdfed` commands, type the command:

```
hdfed -help
```

While you are using the editor, you receive the prompt:

```
he>
```

You can now enter `hdfed` commands.

Many `hdfed` commands have qualifiers, or flags. For instance the info command may optionally be followed by `-all, -long, -group,` or `-label.`

All of the commands and flags can be abbreviated to the extent that their abbreviations are unique. For example `-he i`s an ambiguous abbreviation because it could stand for either the flag `-hexadecimal` or the flag `-help;` on the other hand, the flag `-hel` is not ambiguous.

To obtain information about the usage of a command, type:

```
<command> -help
```

Do not just type the command and expect a help line. Some commands, such as `delete`, do not need any argument, so this could have unfortunate results.

Table 7.2 lists the `hdfed` commands.

Table 7.2 hdfed Commands

Name	Function
help	Provide help on use of `hdfed`
open	Open HDF file
close	Close HDF file
revert	Revert to original file
next	Go to next object/group that satisfies predicate
prev	Go to previous object/group that satisfies predicate
info	Show information about data object
dump	Display data in binary, ASCII, etc.
display	Display a raster image using ICR
put	Put data element as binary into raw file
putr8	Put an ris8 group into raw file
getr8	Get an ris8 group from raw file
delete	Delete an object/group
write	Write an object/group to another HDF file

hdfed: Editing an HDF File

Basics

The `hdfed` utility provides all the capabilities of `hdfls`, but also allows you to examine the data itself and to move data between two HDF files. Actual editing (changing) of individual data objects with hdfed is currently somewhat limited.

`hdfed` is modeled to some extent after `ed`, the Unix line editor. When you invoke `hdfed`, your terminal screen acts as a window into an HDF file. Your initial view of the file is as a set of tags and reference numbers. Each `tag/ref` combination uniquely identifies a data object in the file.

NOTE: The term *data object* in this context refers to the `tag/ref` for the data, plus the data itself. The term *data* or *data element* refers to the actual data that the `tag/ref` refers to. This "data" may be descriptive information, such as a label or dimension record for an image. The term *group* refers to a predefined collection of data objects that correspond to a particular application. For instance, a "raster image group" refers to the collection of data objects that are used to store all of the information in a Raster Image Set.)

Once you have opened an HDF file with `hdfed`, you can do several things with the file, including the following:

- Select an HDF object to examine more closely.
- Move forward or backward among the objects in the file.
- Get information about an object (tag, ref, size, label).
- Display a raster image using the ICR protocol.
- Display a dump of any object.
- Delete an object.
- Annotate an object.
- Write an object to another HDF file.
- Write a data element in its binary form to a non-HDF file.
- Close the file and exit, or open a new file.

`hdfed` also has a small set of special commands, including a conditional statement, a looping statement, an alias command and an unalias command. A simple 'help' feature also exists.

The syntax for hdfed is:

---

hdfed [*filename*] [–nobackup] [–batch]

If `filename` is present, the corresponding file is opened, and a backup is made of the file. Otherwise, a file may be opened from within the editor.

Options

- **- nobackup**
  Specifies that no backup file is to be made. If this option is omitted, a backup file is automatically created.

Table 7.1 Scientific Dataset Routines in the HDF Library (Continued)

hdfseq/hdfrseq	displays sequences of images directly to the screen from HDF files containing raster images.

hdfls: Listing Basic Information about an HDF File

The utility `hdfls` provides a quick look at the tags, reference numbers, and (optionally) lengths of the data elements. The syntax for `hdfls` is:

```
hdfls [-o] [-l] filename
```

-o **Order**: Indicates that the reference numbers are to be displayed in ascending order.

-l **Long format**: Displays more information about the file.

Example 1

A file called `aa.hdf` contains three items associated with a raster image: (1) the image dimensions, (2) a palette, and (3) the raster image. To display information about the contents of the file, enter:

```
hdfls aa.hdf
```

The following is displayed:

```
aa.hdf:
Image Dimensions-8  (Raster-8)  : (tag 200)
  Ref nos: 1
Image Palette-8     (Raster-8)  : (tag 201)
  Ref nos: 3
Raster Image-8      (Raster-8)  : (tag 202)
  Ref nos: 1
```

To display the same information together with the length of each data element, enter:

```
hdfls -l aa.hdf
```

The resulting display is:

```
aa.hdf:
Image Dimensions-8  (Raster-8)  : (tag 200)
      Ref no      1          4 bytes

Image Palette-8     (Raster-8)  : (tag 201)
      Ref no      3        768 bytes

Raster Image-8      (Raster-8)  : (tag 202)
      Ref no      1     120000 bytes
```

Chapter Overview

A small but growing number of utility routines and command line utilities are available for working with HDF files. Currently available programs are described below.

Introduction

The *command line utilities* are application programs that can be executed by entering them at the command level, just like other UNIX commands. These utilities serve two purposes:

1) They make it possible for you to perform, at the command level, common operations on HDF files for which you would normally have to write your own program. For example, the utility `r8tohdf` is a program that takes a raw raster image from a file and stores it in an HDF file in a raster image set.

2) They provide capabilities for doing things with HDF files that would be very difficult to do under your own program control. For example, the utility `hdfseq` takes a a raster image from an HDF file and displays it immediately on a Sun-3 console.

Table 7.1 lists the names and the functions of the utilities described in this chapter. The following sections provide descriptions and examples of these routines.

Table 7.1 Scientific Dataset Routines in the HDF Library

Name	Function
hdfls	displays the tags, ref numbers, and (optionally) lengths of data elements.
hdfed	lets you browse in an HDF file and manipulate some of the data.
fptohdf	converts floating-point data to HDF floating-point data and/or 8-bit raster images.
r8tohdf	converts one or more raw 8-bit images to HDF RIS8 format and writes them to a file, possibly with palettes.
hdftor8	converts images and or palettes from HDF format to raw format and stores them in two corresponding sets of files.
r24hdf8	converts a raw RGB 24-bit image to an 8-bit RIS8 with a palette.
paltohdf	converts a raw palette to hdf format.
hdftopal	converts palette in an hdf file to raw format.

Chapter 7 NCSA HDF Command Line Utilities