fp2hdf

```
fp2hdf -help
fp2hdf <infile> [<in_file>...] -o <out_file> [-r [raster_opts...]|-f]
```

Purpose

Converts floating point data to a scientific data set (SDS) and/or 8-bit raster image set (RIS8) format, storing the results in an HDF file. The image data can be scaled about a mean value.

Description

Input file(s), containing a single two-dimensional or three-dimensional floating point array in either ASCII text, native floating point, or HDF SDS format. 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 maximum and minimum values and/or scales for each axis, these will be used. If the input format is ASCII text or native floating point, it must have the following input fields:

format	"TEXT" or "FP32" or "FP64"
nplanes	"n" for (n+1)dimensional input
nrows	Number of rows in the image
ncols	Number of columns in the image
max_value	Maximum value in the data set
min_value	Minimum value in the data set
plane1 plane2 ...planen	Scales for depth axis
row1 row2 ...rown	Scales for the vertical axis
col1 col2 ...coln	Scales for the horizontal axis
data1 data2 ...datan	Data ordered left to right, top to bottom, front to back.

For FP32 and FP64 input, *format*, *nplanes*, *nrows*, *ncols*, and *nplanes* are native integers; where *format* is the integer representation of the appropriate 4-character string (0x46503332 for FP32 and 0x46503634 for FP64). The remaining input fields are composed of native 32-bit floating point values for FP32 input format, or native 64-bit floating point values for FP64 input format.

Options

-h	Print a helpful summary of usage, and exit.
<*in_file*>	Name(s) of the file(s) containing one 2D or 3D array
-o <*out_file*>	Name of the HDF file to hold the output
-r[raster_opts]|-f	
-r [raster_opts]	Store as a raster image set in the output file
-f	Default to store a SDS in the output file

raster_opts

[-e <*hor*><*vert*>[<*depth*>]|-i <*hor*><*vert*>[<*depth*>] -p <*palfile*> -m <*mean*>

-e <*hor*> <*vert*> [<*depth*>] Expand the data via pixel replication

hor	Number of columns in the new image
vert	Number of rows in the new image
depth	Number of images or depth planes for 3D data only

-i <*hor*> <*vert*> [<*depth*>] Expand the data via interpolation

hor	Number of columns in the new image (> original)
vert	Number of rows in the new image (> original)
depth	Number of images for 3D data only (> original)

-p <*palfile*> Store the palette with the image

palfile	File containing an HDF palette or a raw palette

-m <*mean*> Scale data about a mean floating point value

mean The new extremes (newmax and newmin), as given by:

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

will be equidistant from the mean value. If no mean value is given, then the mean will be:

```
mean=0.5*(max+min)
```

Example Convert floating point data in `f1.txt` to SDS format, and store it in HDF file `o1`:

```
fp2hdf f1.txt -o o1
```

Example Convert floating point data in `f2.hdf` to 8-bit raster format, and store it in HDF file `o2`:

```
fp2hdf f2.hdf -o o2 -r
```

Example Convert floating point data in `f3.bin` to 8-bit raster format and SDS format, and store both in HDF file `o3`:

```
fp2hdf f3.bin -o o3 -r -f
```

Example Convert floating point data in `f4` to a 500x600 raster image, and store the RIS8 in HDF file `o4`. Also store a palette from `palfile` with the image:

```
fp2hdf f4 -o o4 -r -e 500 600 -p palfile
```

Example Convert floating point data in `f5` to 200 planes of 500x600 raster images, and store the RIS8 in HDF file `o5`. Also scale the image data so that it is centered about a mean value of 10.0:

```
fp2hdf f5 -o o5 -r -i 500 600 200 -m 10.0
```

hdf24to8

```
hdf24to8 <in_file> <out_file>
```

Purpose — Converts 24-bit raster image sets into 8-bit raster image sets containing a palette.

Description — The utility **hdf24to8** quantizes a HDF RGB 24-bit image into an 8-bit image with 256-color palette and stores both the palette and image in an HDF file.

Options

in_file	Name of the HDF file containing the 24-bit raster image set
out_file	Name of the HDF file to store the raster image set and palette

Example — All **hdf24to8** operations take the following form:

```
hdf24to8 raster24_file.hdf raster8_file.hdf
```

hdf2jpeg

`hdf2jpeg <in_file> <out_file template>`

Purpose — Copies JPEG compressed images from an HDF file into separate JFIF files.

Description — The **`hdf2jpeg`** utility copies compressed JPEG images from HDF files to JFIF files. If multiple JPEG images are to be copied from the HDF file, the "#" character may be used in the JPEG filename template. The "#" character will be replaced with the reference number of the image being copied. The default JPEG name, if none is specified on the command line is `file.#.jpg`

Options

in_file	Name of HDF file containing the compressed images
out_file	Name of JFIF file to store the JPEG data

Example — To copy one image from an HDF file to a JFIF file, execute a command line similar to the following:

```
hdf2jpeg filename.hdf image1.jfif
```

Example — To copy multiple images from an HDF file to separate JFIF files, execute a command line similar to the following:

```
hdf2jpeg filename.hdf image-#.jfif
```

hdf8to24

```
hdf8to24 [-s <scale>] [-j [quality]] <in_file> <out_file>
```

Purpose — Converts 8-bit raster image sets to 24-bit raster image sets.

Description — The **hdf8to24** utility converts an 8-bit raster image stored in an HDF file into 24-bit raster images stored in a new HDF file. The 8-bit raster image can also be scaled in size during the conversion and stored using JPEG compression.

Options

`-s <scale>`	Magnification option
`<scale>`	Floating point number greater than or equal to 0.01
`-j [quality]`	JPEG compression option with default quality factor of 75
`[quality]`	Integer between 1 and 100, with 100 indicating highest quality
in_file	HDF file containing the 8-bit raster image set
out_file	HDF file containing the 24-bit raster image set

Example — To convert an 8-bit image into a 24-bit image, enter the following command line

```
hdf8to24 file1.hdf file2.hdf
```

Example — To convert an 8-bit image into a 24-bit image scaled by a factor of 2, enter the following command line

```
hdf8to24 -s2 file1.hdf file3.hdf
```

Example — To convert an 8-bit image into a 24-bit compressed image using a JPEG quality factor of 45, enter the following command line

```
hdf8to24 -j45 file1.hdf file4.hdf
```

hdfcomp

```
hdfcomp <out_file> <[-c],[-r],[-i], [-j] <quality>> <in_file> [ ...<[-c],[-r],[-i]>
<in_file>]
```

Purpose

Compresses 8-bit raster images by moving them from one HDF file to another.

Description

The **hdfcomp** utility reads uncompressed 8-bit raster images from HDF files and writes compressed 8-bit raster images to HDF files. Compressed images are appended to the output file, if the output file exists.

Options

out_file Name of the HDF file to store the compressed image

-c|-i|-r|-j Compression type:

- -c Run length encoding
- -i IMCOMP compression (requires HDF file with a palette)
- -r No compression
- -j JPEG compression
  - <*quality*> JPEG quality factor valued from 1 - 100

in_file Name of HDF file containing uncompressed image

Example

A directory contains 20 files named `storm001`, `storm002`, `...storm020`. Each file contains a single 8-bit image. To compress these 20 images using run-length encoding and store them in a file called `allcomp.hdf` execute the following command:

```
hdfcomp allcomp.hdf -c storm*.hdf
```

hdfed

`hdfed [filename] [-nobackup] [-batch]`

Purpose — Edits HDF files at the data element level.

Description — The HDF edit utility **hdfed** is an editor loosely modeled after "ed," the Unix line editor. After invoking **hdfed**, the terminal screen functions as a window into an HDF file. Through this window, you can access the same information provided by **hdfls** or perform rudimentary operations on the data itself. These operations include browsing among elements in a file, displaying images, and dumping, deleting, copying or annotating data. Modifying individual data elements is also possible, however, it is somewhat limited.

The **hdfed** utility is designed for users who need to manipulate HDF files at the data element level. It is not designed to provide a high-level view of HDF files. To use **hdfed** effectively requires familiarity with the format of HDF files, therefore it is not recommended for users who are unfamiliar with low level HDF components. The **hdfed** utility also has a small set of special commands, including a conditional statement, a looping statement, an alias command and an unalias command.

The **hdfed** utility will not allow the abitrary modification of binary data or the data element. It does allow modification of tag and reference numbers within strict constraints. Do not attempt to alter individual bytes. It is acceptable, to replace a data element with another of the same type. See the *HDF User's Guide* for more complete instructions.

Options

filename	Name of the HDF file to edit
-nobackup	Cancels default backup file
-batch	Sets input to a series of commands, rather individual commands
-help	Displays usage information, and a listing of the **hdfed** commands

Commands — Once inside the editor, the -help command may be used after any **hdfed** command (i.e. dump -help) to get more information regarding a specific command.

All of the **hdfed** commands can be abbreviated to the extent that their abbreviations are unique. For example -he is 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. The **hdfed** commands are described as follows:

`annotate [-label|-descriptor] [-editor <editor>]`

Purpose — Edit an annotation

Options

-label	Edit label (default)
-descriptor	Edit descriptor
-editor	Use editor (default EDITOR env value)

`close [-keep]`

Purpose — Close current open file.

Options

-keep	Option to retain backup files.

`delete`

Purpose Delete this object or group.

Options None

`display [-position <xpos> <ypos>] [-expansion <exp>] [-large]`

Purpose Display image on screen.

Options

`-position`	Image position on console screen
`-expansion`	Image expansion factor
`-large`	Make image as large as possible

`dump [-offset <offset>] [-length <len>][-decimal|-short|-byte|-octal|-hexadecimal|-float|-double|-ascii|-udecimal|-ushort]`

Purpose Display the data object contents in the format specified.

Options

`-offset`	Start offset
`-length`	Length to look at
`-decimal`	Decimal format [32 bit integers]
`-short`	Decimal format [16 bit integers]
`-byte`	Decimal format [8 bit integers]
`-octal`	Octal format [Default]
`-hexadecimal`	Hexadecimal format
`-float`	Float format [32 bit floats]
`-double`	Float format [64 bit floats]
`-ascii`	ASCII format
-udecimal	Unsigned decimal format [32-bit floats]
-ushort	Unsigned decimal format [64-bit floats]

`getr8 <`*`filename`*`> <xdim> <ydim> [-pal <palette>] [-raster|-rle|-imcomp]`

Purpose Get an 8-bit raster image set from raw files

Options

`filename`	Name of file containing image
<xdim>	Number of columns in the image
<ydim>	Number of rows in the image
`-pal`	Raw palette file
`-raster`	No compression (default)
`-rle`	Run-length compression
`-imcomp`	IMCOMP compression

`help`

Purpose Print a help screen describing the functionality of **`hdfed`**.

Options None

`info [-all] [-long] [-group] [-label]`

Purpose Show information for one data object.

Options

`-all`	Show info for all objects in file
`-long`	Show more info
`-group`	Organize info in group(s)
`-label`	Show label if any

`next [<predicates>]`

Purpose: Move to the next object that satisfies the predicate

Options: <predicates> are of the general form:
`<item> <comparator> <value>` or `group`

`<item>`	tag, ref, image_size, or label.
`<comparator>`	"=", "!=", ">", "<", ">=", "<="
`<value>`	any number

`open <filename> [-nobackup]`

Purpose: Opens a file *filename*.

`<filename>`	Name of the HDF file to edit
`[-nobackup]`	Cancels default back-up file

`prev [<predicates>]`

Purpose: Move to the next object that satisfies the predicate.

Options: See `next` for examples of `<predicates>`

`put [-file <file>] [-verbose]`

Purpose: Put the raw binary of this object in a file

`-file`	Out file name (default "elt#.@")
`-verbose`	Output diagnostic info

`putr8 [-image <img>] [-palette <pal>] [-verbose]`

Purpose: Put an r8 group into raw image and palette files

`-image`	Image file name (default "img#.@.%")
`-palette`	Palette file name (default "pal#")
`-verbose`	To give output of steps taken

`quit`

Purpose: Exit **`hdfed`**

Options: None

`revert`

Purpose: Discard all changes made in **`hdfed`**.

Options: None

`select [<predicates>] <commands>* end`

Purpose: Step through all elements in the file that satisfy the predicates and execute the commands on them.

Options: if [<predicates>] <commands>* end
Execute commands if predicates are satisfied by element

`wait [<message>]`

Purpose: Print message and then wait for user to hit return

Options: `message`

`write <file> [-attachto <atag> <aref>]`

Purpose: Write an element or group into another HDF file

`-attachto`	What element to write annotation to

hdfls

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

Purpose

Provides a quick listing of the tags, reference numbers, and lengths for each of the data objects contained in the specified file.

Description

The **hdfls** utility lists the tag and reference number for every data element stored in the named file. A tag identifies the type of data element, and the reference number distinguishes a specific data element from the other elements with a similar tag. For example, HDF tags will differentiate raster data from Vdata, but only reference numbers can differentiate one raster element from another.

At this point it is not necessary to identify the purpose of each data element in this file. Note however, that each object is given a different tag and that reference numbers are often repeated for data objects with different tags. It is also important when retrieving data to note that reference numbers are not necessarily assigned in any order. For a more information on the form and function of data objects, their tags, and using reference numbers to retrieve data objects, see the *HDF User's Guide*.

Options

`-o`	Displays the reference numbers in ascending order
`-l`	Displays default information plus the length of each element
`filename`	Name of the HDF file to examine

Example

A file called example.hdf contains three data elements associated with a raster image set: (1) the dimension element, (2) a palette, and (3) the image array. To display the tag and reference number for each element, enter the following command:

```
hdfls example.hdf
```

The information printed to the screen will appear something like this:

```
Example.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: 2
```

hdfpack

```
hdfpack [-i|-b] [-d#] [-t#] <in_file> <out_file>
```

Purpose

Compacts an HDF file by reading in all the data elements in the file and writing them out to a new file.

Description

The **hdfpack** utility condenses HDF files by removing the spaces left behind by deleted data elements. It works by reading the existing data elements from the unpacked file and writing them out to a new or "packed" file.

The **hdfpack** utility also provides a number of additional options for customizing the way HDF files are compacted. To learn more about these options, consult the *HDF User's Guide*.

Options

`-b|-i`	Linked block option
`-b`	Don't coalesce the linked blocks if they exist
`-i`	Prompt for whether or not to coalesce any existing linked blocks
`-d#`	Number of descriptors for all data descriptor lists in the output file
`-t#`	Set the number of linked blocks per table entry in the output file
`in_file`	Name of HDF file containing unpacked data
`out_file`	Name of HDF file containing the packed data

Example

To compact the file `aa.hdf` and put the result in `aa.pck`, enter the following command:

```
hdfpack aa.hdf aa.pck
```

Example

Suppose the file `bb.hdf` contains elements stored as sequences of linked blocks. To compact this file but leave the linked-block elements intact, and put the result in `bb.blk`, enter the following command:

```
hdfpack -b bb.hdf bb.blk
```

hdfrseq

```
hdfrseq [-s] [-l] [-p xloc yloc] [-e magnify] <filename> [...filename]
```

Purpose

Displays one or more 8-bit images in a specified sequence on a Sun-3 or SGI.

Description

The HDF 8-bit remote image sequencer, otherwise known as the **hdfrseq** utility, will display a sequence of images, one after the other, from files containing 8-bit raster image sets. This sequencer will only display images on a Sun console, an SGI IRIS with access to the gllib library, or a Macintosh. If more than one file is listed, the corresponding images will be displayed in the order in which the files are specified. Each new image will overwrite the preceding image on the screen and only one image will be displayed per file.

The **hdfrseq** utility requires a terminal emulator such as NCSA Telnet 2.3 or Teltool 1.2. When **hdfrseq** executes on a file, it converts the image from the HDF 8-bit raster image format into 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. For more information on **hdfrseq**, consult the HDF User's Guide.

Options

`-s`	Display single image until `RETURN` is pressed
`-l`	Paints the image as large as possible on the screen
`-p xloc yloc`	Set the location of image on the screen
xloc	Column number for upper left pixel
yloc	Row number for upper right pixel
`-e magnify`	Enlarges all the images in the sequence
magnify	Number of times to increase image
filename	Name(s) of the file(s) contributing to the image sequence

Example

Assume four images are stored as raster image sets in the files img1.hdf, img2.hdf, img3.hdf, and img4.hdf. To view all four images three times their actual size, enter the following command:

```
hdfrseq -e 3 img1.hdf img2.hdf img3.hdf img4.hdf
```

The images printed to the screen will appear as a four image animation sequence.

hdftopal

```
hdftopal <in_file< <out_file>
```

Purpose — Converts palettes stored in HDF files to palettes stored in raw data files.

Description — The **hdftopal** utility 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. Although HDF stores palettes by interlacing their color components, the raw file will contain a palette stored as 256 red components, followed by 256 green components, and finally 256 blue components.

Options

in_file	HDF file containing a palette
out_file	768-byte output file containing the RGB components

Example — A typical **hdftopal** command line operation appears as follows:

```
hdftopal palette_file.hdf palette_file.raw
```

hdftor8

```
hdftor8 <in_file> [-i] [-v] [-r out_rfile][-p out_pfile]
```

Purpose

Extracts raster image sets from HDF files, separates them into image data and palette data, and saves them to separate raw data files.

Description

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.

Keep in mind that when converting from HDF to raw data files the dimension record is not stored with the image data. As a result, it's best to make a note (mental or otherwise) of the image dimensions before deleting any HDF files.

The *in_file* parameter is the filename for the file to extract images and palettes from, and the *out_rfile* and *out_pfile* are the basic names of the files that will contain the raw images and palettes. These basic 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

in_file	Name of the HDF file containing the raster image set
-i	Sets mode so filenames can be specified interactively
-v	Sets verbose mode to provide descriptive messages
-r *out_rfile*	Specifies the name of the file to store the raster image
-p *out_pfile*	Specified the name of the file to store the palette

Example

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 the following command:

```
hdftor8 denm.hdf
```

Six files are produced with the names `img.1.512.256`, `img.2.512.256`, `img.3.512.256`, `pal.1`, `pal.2`, and `pal.3`.

jpeg2hdf

```
jpeg2hdf <in_file> <out_file>
```

Purpose — Copies a JPEG compressed image from a JFIF compliant file into an HDF file.

Description — The **jpeg2hdf** utility copies a JPEG compressed image from a JFIF compliant file into an HDF file. The image is not decompressed during the copy. If the HDF file already exists, the image is appended to it without overwriting any data.

Options

in_file	File containing the raw raster image
out_file	The HDF file to store the compressed raster image set

Example — To copy several JPEG files into a single HDF file, execute command lines similar to the following:

```
jpeg2hdf image1.jfif store.hdf
jpeg2hdf image2.jfif store.hdf
jpeg2hdf image3.jfif store.hdf
```

paltohdf

`paltohdf <in_file> <out_file>`

Purpose — Converts raw palette files into the HDF format.

Description — The **paltohdf** utility 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 in an RGB RGB ...RGB sequence. This is the standard HDF format for 8-bit palettes.

Options

in_file	768-byte raw data file containing the RGB components
out_file	The HDF file to store the palette

Example — To convert a raw palette into an HDF palette using **paltohdf**, execute a command line similar to:

```
paltohdf palette_file.raw palette_file.hdf
```

r24hdf8

`r24hdf8 <cols> <rows> <in_file> <out_file>`

Purpose

Converts 24-bit images into 8-bit raster image sets containing palettes.

Description

The **r24hdf8** utility converts 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 raw raster file is assumed to contain 8-bit color component values ordered in the following manner: RGB, RGB, ...RGB. The order of pixels is left to right and top to bottom.

The *out_file* is an HDF file with one 8-bit raster image set; i.e. 8-bit image, dimensions, and RGB palette.

Options

cols	Number of columns in the image
rows	Number of rows in the image
in_file	File containing the raw 24 bit image
out_file	The HDF file to store the 8-bit raster image set

Example

To convert a 24-bit image to an 8-bit raster image set, execute a command line similar to the following:

```
r24hdf8 cols rows raster24_file.raw raster8_file.hdf
```

r8tohdf

r8tohdf <*rows*> <*cols*> <*out_file*> [-p *palfile*] [-c|-i|-r] <*in_file*> [...<*in_file*>]

Purpose — Converts raw 8-bit raster images into HDF raster image sets.

Description — The **r8tohdf** utility converts each named raw 8-bit raster image into an HDF raster image set and stored them in a single HDF file. To convert a series of images in a single command line operation, all the named images must be the same size.

Options

rows	Height of the image(s)
cols	Width of the image(s)
out_file	Name of the HDF file that will store the raster image sets
-p *palfile*	Adds the palette to each raster image group
palfile	Name of the file containing the palette
-c|-i|-r	Compression:
-c	Run length encoding
-i	IMCOMP compression
-r	No compression
in_file	Name of the file(s) that contain the raw 8-bit image data

Example — Consider a 256 row x 512 column raw raster image stored in a file called `raster.raw`. Assume its palette is stored in raw form in a file called `palette.raw`. To create a single raster image set by including data from both files, use the following command line operation:

```
r8tohdf 256 512 raster_set.hdf -p palette.raw raster.raw
```

Example — A combination of different types of raster image sets is to be stored in a file called `Example.hdf`. The raster image set created from the first file `Example1.raw` will not contain a palette, however the raster image sets created from `Example2.raw` and `Example3.raw` will include a palette stored in the `pal.raw` file. In addition, `Example1.raw` and `Example2.raw` will be compressed using run length encoding, yet `Example3.hdf` will remain uncompressed. All images are 256 x 512. Performing this conversion will require a command line containing the following information:

```
r8tohdf 256 512 Example.hdf -c Example1.raw -p pal.raw
Example2.raw -r Example3.raw
```

Example — 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 — An 800 x 1000 raw raster is stored in a file called `bigpic`. To convert this image to a run lenght encoded RIS8 stored without a palette in a file called `bigpic.hdf` enter:

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

Example

Three 300 x 400 raw images are contained in files `pic1`, `pic2`, and `pic3`. To convert all three files to raster image sets with imcomp compression and store them in `pic.hdf`, enter:

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

Example

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
```

ristosds

```
ristosds <in_file> [ ...in_file] <-o out_file>
```

Purpose

Converts a series of HDF raster image files into a single HDF scientific data set.

Description

The utility converts a series of HDF raster image files into a single HDF file containing a 3 dimensional scientific data set. Each input file may contains one or more raster images. If there is a palette associated with the images, the palette should be included in the first HDF file. Any subsequent palettes will be ignored.

The resulting scientific data set is a 3-dimensional array whose width and height are equal to the dimensions of the raster images. Consequently, all raster images in a series must have the same dimensions. The third dimension of the array, or its length, is determined by the total number of images included in the conversion.

Options

in_file	Name of HDF file(s) containing the 8-bit raster images set(s)
out_file	Name of HDF file that will contain the 3D scientific data set

Example

Consider a directory containing twenty HDF files named `storm001`, `storm002`, `...storm020`. Assume each file contains a single raster image set with a 100x200 image. To combine this series of images into one scientific data set, execute the following command line:

```
ristosds storm*.hdf -o storm.hdf
```

vcompat

```
vcompat <filename>
```

Purpose — Converts v1.0 Vdata or Vgroup files to v2.0 Vdata or Vgroup files.

Description — This utility modifies HDF files containing Vdata or Vgroup to be compatible with version 2.0 or later of the HDF Vset library. Note that the file is assumed to be incompatible to begin with. This routine will not check to see if the file is already compatible, but it is harmless to run an already-compatible file through again.

Options

filename	Name of the HDF file to be converted

vmake

```
vmake -help
```

Purpose — The **vmake** utility is a general utility for generating Vdata and Vgroups in HDF files. Invoking **vmake** with no arguments provides help. You may use **vmake** in three different ways:

1. to create new Vgroups,
2. to store data in new Vdatas, and
3. to create links among Vgroups and Vdatas.

```
vmake <filename> <Vgroup_name>
```

Description — The **vmake** utility creates a new Vgroup, named *Vgroup_name* in the HDF file called *filename*. This new Vgroup will not overwrite any information within the file, if that file exists. If it doesn't, **vmake** creates a new file. Upon successful creation, the program prints the reference number of that Vgroup.

Options

filename	Name of the HDF file
Vgroup_name	Name of the Vgroup to be created

```
vmake <filename> <Vdata_name> <field=[#][f|b|s|l|c]> [...<field=[#][f|b|s|l|c]>]
```

Description — The **vmake** utility creates a new Vdata, named *Vdata_name* in the HDF file called *filename*, and stores ASCII data from standard input in the Vdata according to field specifications in *format*.

Options

filename	Name of the HDF file
Vdata_name	Name of the Vdata to be created
field=[#][f|b|s|l|c]	
field	Name of the first field in the Vdata record
#	Optional number of components in the field, default is 1
[f|b|s|l|c]	Number type for all components in the field
f	float32
b	int8 (byte)
s	int16 (short)
l	int32 (long)
c	character

Example — This example demonstrates how to create a Vdata named `coordinates` containing 3 fields `PX`, `PY` and `PZ`, each of type 32-bit float. In this case, a stream of ASCII floating-point numbers is read from the ASCII file `pos.dat`.

```
vmake vgroup1.hdf "coordinates" "PX=f,PY=f,PZ=f" < pos.dat
```

Example

This example will create a Vdata named `simulation5` containing a single field of type float named `DENSITY`. In this case, the output from the program `myprog` (a stream of ASCII floating-point numbers) is piped into **vmake**.

```
myprog | vmake vset.hdf "simulation5" "DENSITY=f"
```

vmake <*filename*> <-l> <Vgroup-ref> [...*ref1 ref2* ...*refn*]

Description

The **vmake** utility in this context allows one or several Vdatas or Vgroups to be inserted (linked) to a Vgroup. All Vgroups and Vdata are identified by their reference numbers. Reference numbers of existing Vgroups and Vdatas can be determined through **vshow**.

The **vmake** utility permits the creation of empty Vgroups, but not empty Vdatas. Vgroups and Vdatas may be created in any order by the user. However, to perform linking, the Vgroups and Vdatas involved must already exist. If **vmake** is successful it prints a positive number; otherwise, a negative number is printed.

The **vmake** utility does not check for cycles, and will happily allow you to insert a Vgroup into itself. It is legal to create multiple Vgroups with identical names.

Options

filename	Name of the HDF file
-l	Indicates use of link option to add objects to a Vgroup
Vgroup_ref	Reference number for the Vgroup into which the other objects are to be inserted
ref1 ref2 ...refn	Reference numbers for objects to be inserted into the Vgroup

vshow

`vshow` *`filename`* `[+[n]]`

Purpose — Lists Vdata and Vgroup information in an HDF file.

Description — For an HDF file containing Vdata or Vgroups, **`vshow`** will display information on all the Vgroups in the file, in the order stored, followed by the Vdata in the file.

Information about Vdata that are not linked to a Vgroup are preceded by an "L".

The **`vshow`** utility is primarily used to examine the contents of a file. But **`vshow`** is also useful for converting Vdata and Vgroups files into ASCII. This utility outputs full data dumps of Vdata through standard output, and outputs general information through standard error.

For full Vdata dumps, specify either one Vdata or all Vdatas.

Options

`n`	dumps data for the Vdata with reference number *`n`*, unformatted.
`+`*`n`*	gives full dump of Vdata with ID n.

Example — The following example will create an ASCII file containing formatted ASCII data from the Vdata with reference number 4.

```
vshow myfile.hdf +4 > text.file
```