HDF4 Examples

HDF User's Guide Examples

These examples are used in the HDF User's Guide. The examples are included in the HDF source code, and the links below are to locations in the unpacked source code.

Annotations (AN):

These examples are used in Chapter 10 -- Annotations (AN API) of the HDF User's Guide.

Example Description	C	Fortran
Creating File and Data Annotations This example illustrates the use of ANcreatef/affcreate to create file annotations and ANcreate/afcreate to create data annotations. In this example, the program creates an HDF file named "General_HDFobjects.hdf" then attaches two annotations to it, a file label and a file description. Within the HDF file, the program creates a vgroup named "AN Vgroup" and attaches two annotations to it, a data label and a data description. Refer to Chapter 5, Vgroups (V API), for a discussion of the V interface routines used in this example. Note that the names AN_FILE_LABEL, AN_FILE_DESC, AN_DATA_LABEL, and AN_DATA_DESC are defined by the library to specify the type of the annotation to be accessed.	AN_create_annotation.c	AN_create_annotation.f
Reading File and Data Annotations This example illustrates the use of ANfileinfo/affileinfo to get the number of data and file annotations in the file. ANselect/afselect gets an annotation, ANannlen/afannlen gets the length of the annotation, and ANreadann/afreadann reads the contents of the annotation. In this example, the program reads some of the annotations created in the file "General_HDFobjects.hdf" by Example 1. The program first gets the information on the annotations in the file so that the number of existing annotations of each kind is available prior to reading. The program then gets the length of each annotation and allocates sufficient space for the contents of the annotation to be read. For the simplicity of this example, only the data labels are read. Any other annotations can be read by adding the for loop with appropriate values as noted below.	AN_read_annotation.c	AN_read_annotation.f
Obtaining Annotation Information This example illustrates the use of ANnumann/afnumann to obtain the number of annotations of an object, ANannlist/afannlist to obtain the list of annotation identifiers, and ANid2tagref/afidtagref, ANatype2tag/afatypetag, and ANtag2atype/aftagatype to perform some identifier conversions. In this example, the program locates the vgroup named "AN Vgroup" that was created in the file "General_HDFobjects.hdf" by Example 1. The program then gets the number of data descriptions that this vgroup has and the list of their identifiers. If there are any identifiers in the list, the program displays the corresponding reference numbers. Finally, the program makes two simple conversions, from an annotation type to a tag and from a tag to an annotation type, and displays the results.	AN_get_annotation_info.c	AN_get_annotation_info.f

General Raster (GR):

These examples are used in Chapter 8 -- General Raster Images (GR API) of the HDF User's Guide.

Example Description	C	Fortran
Creating and Writing a Raster Image This example illustrates the use of the routines Hopen/hopen, GRstart/mgstart, GRcreate/mgcreat, GRwriteimage/mgwrimg, GRendaccess/mgendac, GRend/mgend, and Hclose/hclose to create an HDF file and store a raster image in it. In this example, the program creates the HDF file called "General_RImages.hdf" and a raster image in the file. The image created is of size 5x10 and named "Image Array 1", and has data of the int16 data type, 2 components, and interlace mode MFGR_INTERLACE_PIXEL. Then the program writes the image data, terminates access to the image and the GR interface, and closes the file.	GR_create_and_write_image.c	GR_create_and_write_image.f
Modifying an Existing Raster Image This example illustrates the use of the routines GRselect/mgselct to obtain an existing raster image and GRwrite/mgwrimg to modify image data. In this example, the program selects the only raster image in the file "General_RImages.hdf" created and written in Example 1, and modifies image data. The program also creates another raster image that is named "Image Array 2" and has 3 components with dimension size of 4x6, data type of DFNT_CHAR8, and interlace mode of MFGR_INTERLACE_PIXEL.	GR_modify_image.c	GR_modify_image.f
Reading a Raster Image This example illustrates the use of the routine GRreadimage/mgrdimg to read an image and its subsets. In this example, the program reads the image written by Example 1 and modified by Example 2 in the file "General_RImages.hdf". Recall that this image has two components and has 5 rows and 10 columns. The program first reads the entire image, then reads a subset of the image, 3 rows and 2 columns starting at the 2nd row and the 4th column, and finally reads the image skipping all the even rows and all the odd columns. Reading patterns are applied to all components.	GR_read_image.c	GR_read_image.f
Obtaining File and Image Information This example illustrates the use of the routines GRfileinfo/mgfinfo and GRgetiminfo/mggiinf to obtain information such as the number of images and attributes in an HDF file and the characteristics of a raster image in the file. In this example, the program gets the number of images in the file using the routine GRfileinfo/mgfinfo. For each image, the program then obtains and displays its name, number of components, data type, interlace mode, dimension sizes, and number of attributes using the routine GRgetiminfo/mggiinf.	GR_image_info.c	GR_image_info.f
Operations on File and Raster Image Attributes This example illustrates the use of the routines GRsetattr/mgsnatt/mgscatt to assign attributes to an HDF file and to an image. In this example, the program sets two attributes to the existing file "General_RImages.hdf" and two attributes to the image named "Image Array 2". The file is created by the program in Example 1 and the image is created by the program in Example 2. The values of the second attribute of the image are of type int16 and the values of the other three attributes are of type char8.	GR_set_attribute.c	GR_set_attribute.f

Obtaining File and Image Attribute This example illustrates the use of the routines GRattrinfo/mgatinf, GRfindattr/mgfndat, and GRgetattr/mggnatt/mggcatt to extract information and values of file and image attributes that were set by the program in Example 5. In this example, the program gets the information about each file attribute, then extracts its values. The program then selects the second image in the file, finds the attribute named "Image Attribute 2", obtains the data type and the number of values in the attribute, and extracts its stored values.	GR_get_attribute.c	GR_get_attribute.f
Writing a Palette This example illustrates the use of the routines GRgetlutid/mggltid and GRwritelut/mgwclut to attach a palette to a raster image and write data to it. In this example, the program creates an image named "Image with Palette" in the file "Image_with_Palette.hdf". A palette is then attached to the image and data is written to it.	GR_write_palette.c	GR_write_palette.f
Reading a Palette This example illustrates the use of the routines GRgetlutinfo/mgglinf and GRreadlut/mgrclut to obtain information about a palette and to read palette data. In this example, the program finds and selects the image named "Image with Palette" in the file "Image_with_Palette.hdf". Then the program obtains information about the palette and reads the palette data.	GR_read_palette.c	GR_read_palette.f
Creating and Writing a Chunked Raster Image This example illustrates the use of the routines Hopen/hopen, GRstart/mgstart, GRcreate/mgcreat, GRwritechunk/mgwchnk, GRendaccess/mgendac, GRend/mgend, and Hclose/hclose to create an HDF file and store a raster image in it. In this example, the program creates an image of 6 rows by 10 columns in C and 10 rows by 6 columns in FORTRAN. The image is set up to be chunked with a chunk size of 3x2 in C and 2x3 in FORTRAN and compressed with the GZIP method. Three chunks are then written to the image.	GR_write_chunks.c	

Scientific Datasets (SD):

These examples are used in Chapter 3 --Scientific Data Sets (SD API) of the HDF User's Guide.

Example Description	C	Fortran
Creating an HDF file and an Empty SDS This example illustrates the use of SDstart/sfstart, SDcreate/sfcreate, SDendaccess/sfendacc, and SDend/sfend to create the HDF file named SDS.hdf, and an empty data set with the name SDStemplate in the file. Note that the Fortran program uses a transformed array to reflect the difference between C and Fortran internal data storages. When the actual data is written to the data set, SDS.hdf will contain the same data regardless of the language being used.	SD_create_sds.c	SD_create_sds.f

Writing to an SDS This example illustrates the use of the routines SDselect/sfselect and SDwritedata/sfwrite to select the first SDS in the file SDS.hdf created in Example 1 and to write actual data to it.	SD_write_to_sds.c	SD_write_to_sds.f
Writing a Slab of Data to an SDS This example shows how to fill a 3-dimensional SDS array with data by writing a series of 2-dimensional slabs to it.	SD_write_slab.c	SD_write_slab.f
Altering Values within an SDS Array This example demonstrates how the routine SDwritedata can be used to alter the values of the elements in the 10th and 11th rows, at the 2nd column, in the SDS array created in the Example 1 and written in Example 2. The FORTRAN-77 routine sfwdata is used to alter the elements in the 2nd row, 10th and 11th columns, to reflect the difference between C and Fortran internal storage.	SD_alter_sds_values.c	SD_alter_sds_values.f
Appending Data to an SDS Array with an Unlimited Dimension This example creates a 10x10 SDS array with one unlimited dimension and writes data to it. The file is reopened and the routine SDisrecord/sfisrcrd is used to determine whether the selected SDS array is appendable. Then new data is appended, starting at the 11th row.	SD_unlimited_sds.c	SD_unlimited_sds.f
Compressing SDS Data This example uses the routine SDsetcompress/sfscompress to compress SDS data with the GZIP compression method. See comments in the program regarding the use of the Skipping Huffman or RLE compression methods.	SD_compress_sds.c	SD_compress_sds.f
Moving Data to an External File This example illustrates the use of the routine SDsetexternalfile/sfsextf to move the SDS data written in Example 2 to an external file.	SD_mv_sds_to_external.c	SD_mv_sds_to_external.f
Reading from an SDS This example uses the routine SDreaddata/sfrdata to read the data that has been written in Example 2, modified in Example 4, and moved to the external file in Example 7. Note that the original file SDS.hdf that contains the SDS metadata and the external file ExternalSDS that contains the SDS raw data should reside in the same directory. The fact that raw data is in the external file is transparent to the user's program.	SD_read_from_sds.c	SD_read_from_sds.f
Reading Subsets of an SDS This example shows how parameters start, stride, and edges of the routine SDreadata/sfrdata can be used to read three subsets of an SDS array. C: For the first subset, the program reads every 3rd element of the 2nd column starting at the 4th row of the data set created in Example 1 and modified in Examples 2, 4 and 7. For the second subset the program reads the first 4 elements of the 10th row. For the third subset, the program reads from the same data set every 6th element of each column and 4th element of each row starting at 1st column, 3d row. FORTRAN-77: Fortran program reads transposed data to reflect the difference in C and Fortran internal storage.	SD_read_subsets.c	SD_read_subsets.f

Getting Information about a File and SDS This example illustrates the use of the routine SDfileinfo/sffinfo to obtain the number of data sets in the file SDS.hdf and the routine SDgetinfo/sfginfo to retrieve the name, rank, dimension sizes, data type and number of attributes of the selected data set.	SD_get_info.c	SD_get_info.f
Locating an SDS by Its Name This example uses the routine SDnametoindex/sfn2index to locate the SDS with the specified name and then reads the data from it.	SD_find_sds_by_name.c	SD_find_sds_by_name.f
Setting and Retrieving Dimension Information This example illustrates the use of the routines SDgetdimid/sfdimid, SDsetdimname/sfsdmname, SDsetdimscale/sfsdscale, SDdiminfo/sfgdinfo, and SDgetdimscale/sfgdscale to set and retrieve the dimensions names and dimension scales of the SDS created in Example 1 and modified in Examples 2, 4 and 7.	SD_set_get_dim_info.c	SD_set_get_dim_info.f
Distinguishing a Dimension Scale from a Data Set in a File This example illustrates the use of the routine SDiscoordvar/sfiscvar to determine whether the selected SDS array is a data set or a dimension stored as an SDS array (coordinate variable) (see discussion in Section 3.8.4) and displays the name of the data set or dimension.	SD_dimscale_vs_sds.c	SD_dimscale_vs_sds.f
Setting Attributes This example shows how the routines SDsetattr/sfscatt/sfsnatt are used to set the attributes of the file, data set, and data set dimension created in the Examples 2, 4, and 12.	SD_set_attr.c	SD_set_attr.f
Reading Attributes This example uses the routines SDfindattr/sffattr, SDattrinfo/sfgainfo, and SDreadattr/sfrattr to find and read attributes of the file, data set, and data set dimension created in the Example 14.	SD_get_attr.c	SD_get_attr.f

Calibrating Data
 	
Suppose the values in the calibrated array cal_val are the following integers:
`cal_val[6] = {2, 4, 5, 11, 26, 81}`
By applying the calibration equation orig = cal * (cal_val - offset) with cal = 0.50 and offset = -2000.0, the calibrated array cal_val[] returns to its original floating-point form:
`original_val[6] = {1001.0, 1002.0, 1002.5, 1005.5, 1013.0, 1040.5}`
	N/A	N/A

Writing and Reading a Chunked SDS This example demonstrates the use of the routines SDsetchunk/sfschnk, SDwritedata/sfwdata, SDwritechunk/sfwchnk, SDgetchunkinfo/sfgichnk, SDreaddata/sfrdata, and SDreadchunk/sfrchnk to create a chunked data set, write data to it, get information about the data set, and read the data back. Note that the Fortran example uses transposed data to reflect the difference between C and Fortran internal storage.	SD_chunking_example.c	SD_chunking_example.f

Vdatas (VD):

These examples are used in Chapter 4 -- Vdatas (VS API) of the HDF User's Guide.

Example Description	C	Fortran
Accessing a Vdata in an HDF File This example illustrates the use of Hopen/hopen, Vstart/vfstart, VSattach/vsfatch, VSdetach/vsfdtch, Vend/vfend, and Hclose/hclose to create and to access different vdatas from different HDF files. The program creates an HDF file, named "General_Vdatas.hdf", containing a vdata. The program also creates a second HDF file, named "Two_Vdatas.hdf", containing two vdatas. Note that, in this example, the program does not write data to these vdatas. Also note that before closing the file, the access to its vdatas and its corresponding Vdata interface must be terminated. These examples request information about a specific vdata.	VD_create_vdatas.c	VD_create_vdatas.f
Creating and Storing One-field Vdatas Using VHstoredata and VHstoredatam This example illustrates the use of VHstoredata/vhfscd and VHstoredatam/vhfsdm to create single-field vdatas. This example creates and writes two vdatas to the file "General_Vdatas.hdf". The first vdata is named "First Vdata", contains 5 records, and belongs to a class named "5x1 Array". The second vdata is named "Second Vdata", contains 6 records, and belongs to a class named "6x4 Array". The field of the first vdata is a single-component field, i.e., order of 1, and named "Single-component Field". The field of the second vdata has an order of 4 and is named "Multi-component Field". In these examples two vdatas are created. The first vdata has five records with one field of order 1 and is created from a 5 x 1 array in memory. The second vdata has six records with one field of order 4 and is created from a 6 x 4 array in memory.	VD_create_onefield_vdatas.c	VD_create_onefield_vdatas.f
Writing a Vdata of Homogeneous Type This example illustrates the use of VSfdefine/vsffdef, VSsetname/vsfsnam, VSsetclass/vsfscls, VSsetfields/vsfsfld, and VSwrite/vsfwrt to create and write a three-field vdata to the file "General_Vdatas.hdf". Although the fields have data of the same type, they have different orders. To clarify the illustration, let us assume that the vdata is used to contain the data of some particles collected from an experiment. Each record of the data includes the position of a particle, its weight, and the minimum and maximum temperature the particle can endure. The vdata is named "Solid Particle", contains 10 records, and belongs to a class, named "Particle Data". The fields of the vdata include "Position", "Mass", and "Temperature". The field "Position" has an order of 3 for the x, y, and z values representing the position of a particle. The field "Mass" has an order of 1. The field "Temperature" has an order of 2 for the minimum and maximum temperature. The program creates the vdata, sets its name and class name, defines its fields, and then writes the data to it.	VD_write_to_vdata.c	VD_write_to_vdata.f

Writing a Multi-field and Mixed-type Vdata with Packing This example illustrates the use of VSfpack/vsfnpak/vsfcpak and VSwrite/vsfwrit to write a vdata with data of different types. Note that the approach used in Example 3 makes it difficult for the vdata to have mixed-type data. In this example, the program creates an HDF file, named "Packed_Vdata.hdf", then defines a vdata which is named "Mixed Data Vdata" and belongs to class "General Data Class". The vdata contains four order-1 fields, "Temp", "Height", "Speed", and "Ident" of type float32, int16, float32, and char8, respectively. The program then packs the data in fully interlaced mode into the buffer "databuf" and writes the packed data to the vdata. Note that, in the C example, a VSfpack call packs all N_RECORDS and a VSwrite call writes out all N_RECORDS records. In the Fortran example, N_RECORDS of each field are packed using separate calls to vsfnpak and vsfcpak; vsfwrit writes packed data to the vdata.	VD_write_mixed_vdata.c	VD_write_mixed_vdata.f
Reading a Vdata of Homogeneous Type This example illustrates the use of VSfind/vsffnd to locate a vdata given its name, VSseek/vsfseek to move the current position to a desired record, and VSread/vsfrd to read the data of several records. The function VSfind will be discussed in Section 4.7.3. The approach used in this example can only read data written by a program such as that in Example 3, i.e., without packing. Reading mixed data vdatas must use the approach illustrated in Example 6. The program reads 5 records starting from the fourth record of the two fields "Position" and "Temperature" in the vdata "Solid Particle" from the file "General_Vdatas.hdf". After the program uses VSfind/vsffnd to obtain the reference number of the vdata, it uses VSseek/vsfseek to place the current position at the fourth record, then starts reading 5 records, and displays the data.	VD_read_from_vdata.c	VD_read_from_vdata.f
Reading a Multi-field and Mixed-type Vdata with Packing This example illustrates the use of VSread/vsfread to read part of a mixed data vdata and VSfpack/vsfnpak/vsfcpak to unpack the data read. The program reads the vdata "Mixed Data Vdata" that was written to the file "Packed_Vdata.hdf" by the program in Example 4. In Example 6, all values of the fields "Temp" and "Ident" are read. The program unpacks and displays all the values after reading is complete. Again, note that in C only one call to VSread and one call to VSfpack are made to read and unpack all N_RECORDS records. In Fortran, data is read with one call to vsfread, but each field is unpacked using separate calls to vsfnpak and vsfcpak.	VD_read_mixed_vdata.c	VD_read_mixed_vdata.f
Locating a Vdata Containing Specified Field Names This example illustrates the use of VSgetid/vsfgid to obtain the reference number of each vdata in an HDF file and the use of VSfexist/vsfex to determine whether a vdata contains specific fields. In this example, the program searches the HDF file "General_Vdatas.hdf" to locate the first vdata containing the fields "Position" and "Temperature". The HDF file is created in Example 1 and modified in Examples 2 and 3.	VD_locate_vdata.c	VD_locate_vdata.f

Operations on Field and Vdata Attributes This example illustrates the use of VSsetattr/vsfscat/vsfsnat to attach an attribute to a vdata and to a field in a vdata, the use of VSattrinfo/vsfainf to get information about a field attribute and a vdata attribute, and the use of VSgetattr/vsfgcat/vsfgnat to get the values of an attribute of a vdata and the values of an attribute of a field in a vdata. The example also shows the use of VSfnattrs/vsffnas to obtain the number of attributes attached to a field of a vdata and the use of VSnattrs/vsfnats to obtain the total number of attributes attached to both a vdata and its fields. In this example, the program finds the vdata, named "Solid Particle", in the HDF file "General_Vdatas.hdf" is created in Example 1 and modified in Examples 2 and 3. It then obtains the index of the field, named "Mass", in the vdata. An attribute named "Site Ident" is attached to the vdata to contain the identification of the experiment sites. Another attribute named "Scales" is attached to the field for its scale values. The vdata attribute has 3 character values and the field attribute has 4 integer values.	VD_set_get_vdata_attr.c	VD_set_get_vdata_attr.f
Obtaining Vdata Information This example illustrates the use of VSgetid/vsfgid and VSinquire/vsfinq to obtain information about all vdatas in an HDF file. In this example, the program uses VSgetid to locate all vdatas in the HDF file "General_Vdatas.hdf", which is created in Example 1 and modified in Examples 2 and 3. For each vdata found, if it is not the storage of an attribute, the program uses VSinquire/vsfinq to obtain information about the vdata and displays its information. Recall that an attribute is also stored as a vdata; the function VSisattr/vsfisat checks whether a vdata is a storage of an attribute.	VD_get_vdata_info.c	VD_get_vdata_info.f

Vgroups (VG):

These examples are used in Chapter 5 -- Vgroups (V API) of the HDF User's Guide.

Example Description	C	Fortran
Creating HDF Files and Vgroups This example illustrates the use of Hopen/hopen, Vstart/vfstart, Vattach/vfatch, Vdetach/vfdtch, Vend/vfend, and Hclose/hclose to create and to access two vgroups in an HDF file. The program creates the HDF file, named "Two_Vgroups.hdf", and two vgroups stored in the file. Note that, in this example, the program only create two empty vgroups.	VG_create_vgroup.c	VG_create_vgroup.f
Adding an SDS to a New Vgroup This example illustrates the use of Vaddtagref/vfadtr to add an HDF data object, an SDS specifically, to a vgroup. In this example, the program first creates the HDF file "General_Vgroups.hdf", then an SDS in the SD interface, and a vgroup in the Vgroup interface. The SDS is named "Test SD" and is a one-dimensional array of type int32 of 10 elements. The vgroup is named "SD Vgroup" and is of class "Common Vgroups". The program then adds the SDS to the vgroup using Vaddtagref/vfadtr. Notice that, when the operations are complete, the program explicitly terminates access to the SDS, the vgroup, the SD interface, and the Vgroup interface before closing the HDF file. Refer to Chapter 3, Scientific Data Sets (SD API) for the discussion of the SD routines used in this example.	VG_add_sds_to_vgroup.c	VG_add_sds_to_vgroup.f

Adding Three Vdatas into a Vgroup This example illustrates the use of Vinsert/vfinsrt to add a vdata to a vgroup. Note that Vaddtagref/vfadtrf, used in the previous example, performs the same task and only differs in the argument list. In this example, the program creates three vdatas and a vgroup in the existing HDF file "General_Vgroups.hdf" then adds the three vdatas to the vgroup. Notice that the vdatas and the vgroup are created in the same interface that is initialized by the call Vstart/vfstart. The first vdata is named "X,Y Coordinates" and has two order-1 fields of type float32. The second vdata is named "Temperature" and has one order-1 field of type float32. The third vdata is named "Node List" and has one order-3 field of type int16. The vgroup is named "Vertices" and is of class "Vertex Set". The program uses Vinsert/vfinsrt to add the vdatas to the vgroup using the vdata identifiers. Refer to Chapter 4, Vdatas (VS API), for the discussion of the VS routines used in this example.	VG_insert_vdatas_to_vgroup.c	VG_insert_vdatas_to_vgroup.f
Obtaining Information about Lone Vgroups This example illustrates the use of Vlone/vflone to obtain the list of reference numbers of all lone vgroups in the file and the use of Vgetname/vfgnam and Vgetclass/vfgcls to obtain the name and the class of a vgroup. In this example, the program calls Vlone/vflone twice. The first call is to obtain the number of lone vgroups in the file so that sufficient space can be allocated; the later call is to obtain the actual reference numbers of the lone vgroups. The program then goes through the list of lone vgroup reference numbers to get and display the name and class of each lone vgroup. The file used in this example is "General_Vgroups.hdf".	VG_get_vgroup_info.c	VG_get_vgroup_info.f
Operations on Vgroup Attributes This example illustrates the use of Vfind/vfind to locate a vgroup by its name, Vsetattr/vfscatt to attach an attribute to the vgroup, Vattrinfo/vfainfo to obtain information about the vgroup attribute, and Vgetattr/vfgcatt to obtain the attribute values. The program obtains the version of the group then sets an attribute named "First Attribute" for the vgroup named "SD Vgroup". Next, the program gets the number of attributes that the vgroup has, and obtains and displays the name, the number of values, and the values of each attribute.	VG_set_get_vgroup_attr.c	VG_set_get_vgroup_attr.f
Obtaining Information about the Contents of a Vgroup This example illustrates the use of Vgetid/vfgid to get the reference number of a vgroup, Vntagrefs/vfntr to get the number of HDF data objects in the vgroup, Vgettagref/vfgttr to get the tag/reference number pair of a data object within the vgroup, and Visvg/vfisvg and Visvs/vfisvs to determine whether a data object is a vgroup and a vdata, respectively. In the example, the program traverses the HDF file "General_Vgroups.hdf" from the beginning and obtains the reference number of each vgroup so it can be attached. Once a vgroup is attached, the program gets the total number of tag/reference number pairs in the vgroup and displays some information about the vgroup. The information displayed includes the position of the vgroup in the file, the tag/reference number pair of each of its data objects, and the message stating whether the object is a vdata, vgroup, or neither.	VG_vgroup_contents.c	VG_vgroup_contents.f

Other Examples

Example of Converting Remotely Sensed Data into Chunked HDF (4) Files:

Example	Contents	Compressed tar file
ChunkBinary This example converts a 6144 x 6144 array of 16 bit unsigned ints into a chunked HDF scientific data set. Since a 6144x6144 array is very large the array must be converted by reading in the array in sections, a smaller array or set of scanlines. If your machine does not have enough memory to read in the first set of scanlines, the chunkbinary program will subdivide the array into smaller fractions and try to read in that smaller array into memory.	Contents of ChunkBinary	Compressed tar file of ChunkBinary example
Chunkit This example converts PATHFINDER Sea Surface Temperature (SST) data in HDF format into chunked HDF files. Chunkit reads from one PATHFINDER SST input file and writes to a new file containing the new chunked datasets.	Contents of Chunkit	Compressed tar file of Chunkit example