Chandra X-ray Center

URL: http://cxc.harvard.edu/ciao3.4/writefits.html
Last modified: December 2006

AHELP for CIAO 3.4

writefits

Context: varmm

Jump to: Description Examples CHANGES IN CIAO 3.1 Bugs See Also

Synopsis

S−Lang function to create a FITS output file.

Syntax

```
Integer_Type writefits( filename, data )
Integer_Type writefits( filename, data, extname )
Integer_Type writefits( filename, data, extname, key_array )
Integer_Type writefits( filename, data, extname, key_array, clobber )

Error Return Value: 0 on success and −1 on failure.
```

Description

The writefits() function generates FITS output files from S−Lang structured variables. FITS images or binary tables may be created, using input data in either CIAO or ISIS format (see below). Optional arguments include a string array of FITS−legal header keywords, as well as a file clobber boolean (which defaults to 0, ie false).

The data parameter is a structured S−Lang variable in either S−Lang/ISIS or CIAO/Varmm formats. The former is just a collection of raw column fields (or a pixel array) within a structure, while the latter augments the raw data with metadata (denoted by fields beginning with underscores) describing the originating path and filename, applied filter, etc (see "ahelp readfile" for further information). Within the routine, 'data' will be interpreted as follows: if it

- is a single, unnamed N−dim array OR contains a named '_pixels' array field, then it will be written to the file as an image
- contains a named '_nrows' field then all of the fields following 'nrows' will be written as cols within a binary table
- does not contain _nrows but rather some other series of named fields, then those fields will be written as columns to a binary table
- if a non−NULL header key string is passed in, write this as the header, otherwise use the '_header' field in the structure if it exists.

Example 1

Convert an ASCII file to a FITS file in 2 lines

```
chips> ascii = readfile("ascii.dat")
chips> writefits("copy.fits",ascii,"duplicate")
0
```

The output file will contain a Table block called "duplicate" which contains the data from the input file. If ascii.dat contained two columns with 3 elements each then the output file would look like:

```
unix% dmlist copy.fits blocks

--------------------------------------------------------------------
Dataset: copy.fits
--------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------
Block    1: PRIMARY                        Null
Block    2: duplicate                      Table         2 cols x 3        rows
unix% dmlist copy.fits cols
--------------------------------------------------------------------
Columns for Table Block duplicate
--------------------------------------------------------------------

ColNo  Name                 Unit        Type             Range
   1   col1                              Real4          −Inf:+Inf            label for field   1
   2   col2                              Real4          −Inf:+Inf            label for field   2
```

The ascii2fits tool can also be used for this purpose.

Example 2

Appending to a FITS table from ChIPS/Sherpa

The next exmaple shows a simple way one can copy or rearrange a table within a FITS file. The first writefits call specifies a NULL key array and will clobber swap.fits if present. The second and third calls simply append the blocks to the existing file (the default). Many other paste/merge/sort operations are possible, with similar relative ease.

```
chips> bl1 = readfile("table.fits[2]")
chips> bl2 = readfile("table.fits[3]")
chips> bl3 = readfile("table.fits[4]")
chips> writefits("swap.fits",bl3,"one",NULL,1)
0
chips> writefits("swap.fits",bl2,"two")
0
chips> writefits("swap.fits",bl1,"three")
0
```

Example 3

Appending to a FITS table from slsh

This is a copy of the previous example except that the code is written to a file that can be run with slsh.

```
require("varmm");
variable bl1 = readfile("table.fits[2]");
variable bl2 = readfile("table.fits[3]");
variable bl3 = readfile("table.fits[4]");
() = writefits("swap.fits",bl3,"one",NULL,1);
() = writefits("swap.fits",bl2,"two");
() = writefits("swap.fits",bl1,"three");
```

Since this is a S−Lang script, the Varmm module must be loaded, variables defined, and statements have to end in a semi−colon. We also have to account for the return value of the writefits() function; here we ignore the value but in general it should be checked for success.

Example 4

Writing out an image to a FITS file

As discussed in "ahelp readimage", the axis order used by S−Lang and FITS is different: S−Lang uses the C system where the right−most index loops fastest whilst FITS uses the FORTRAN scheme where the left−most index loops fastest. This means that a 5 by 2 S−Lang array will be stored as a 2 by 5 FITS image, as shown here.

```
chips> a = _reshape( [1:10], [5,2] )
chips> a
Integer_Type[5,2]
chips> print( a )
1       2
3       4
5       6
7       8
9       10
chips> writefits( "temp.img", a )
0
```

This creates the file temp.img which dmlist reports as:

```
unix% dmlist temp.img blocks

--------------------------------------------------------------------
Dataset: temp.img
--------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------
Block    1: PRIMARY                        Image      Int4(2x5)
unix% dmlist temp.img data,array

--------------------------------------------------------------------
Data for Image Block PRIMARY
--------------------------------------------------------------------

ROW    CELL   PRIMARY[2,5]

     1 [     1     1 ]           1
     1 [     2     1 ]           2
```

```
      1 [     1     2 ]          3
      1 [     2     2 ]          4
      1 [     1     3 ]          5
      1 [     2     3 ]          6
      1 [     1     4 ]          7
      1 [     2     4 ]          8
      1 [     1     5 ]          9
      1 [     2     5 ]         10
```

The data has been written out correctly, it is just that the order of the axes has been swapped. If you use readimage() to read in "temp.img" the pixels field of the returned structure will match the original S−Lang array (here the variable a).

```
  chips> b = readimage( "temp.img" )
  Warning: Could not retrieve WCS coord descriptor
  chips> b.pixels
  Integer_Type[5,2]
  chips> print( b.pixels )
  1       2
  3       4
  5       6
  7       8
  9       10
```

CHANGES IN CIAO 3.1

Writing out non−square FITS images

Prior to CIAO 3.1, non−square images were written out by writefits() incorrectly.

Bugs

See the bugs page for the Varmm library on the CIAO website for an up−to−date listing of known bugs.

See Also

modules
    varmm
varmm
    fits_bitpix, readarf, readascii, readbintab, readfile, readimage, readpha, readrdb, readrmf, writeascii

URL: http://cxc.harvard.edu/ciao3.4/writefits.html
Last modified: December 2006