A FITS Library Package

Donald. H. Gudehus,
Georgia State University

Abstract

The Flexible Image Transport System (FITS) Library Package for Linux, and Mac OS X consists of a collection of high and low level routines to carry out various useful functions associated with FITS image files. Besides the source code for the library, a demonstration program, "testfits", which illustrates the library's capabilities and shows how to code the subroutines into your own customized applications, is included. Source code of subroutines from the MIIPS Plot Package and subroutines to display images on Enhanced SAOimage are included as well. The source code, executables, and documentation for Linux and Mac OS X are available at http://www.chara.gsu.edu/~gudehus/fits_library_package.html.

1. Introduction

The Flexible Image Transport System (FITS) (Wells, Greisen, and Harten, 1981) has become a standard format for astronomical image data. Data sets can be organized as N-dimensional arrays, where in the most general case for images, the data would have three axes, representing columns, rows, and planes. On the other hand, a single row of image data could be represented as just one axis. The FITS standard also allows for extensions within a file whereby additional images can appended to the initial image. These images are sometimes referred to as maps. The purpose of creating this FITS Library Package is to provide a set of high level routines that can carry out useful operations on FITS image files, and which are easy to implement in customized programs. The lower level routines upon which the high level ones are based, can themselves be used for various specialized purposes, if desired. The FITS Library is currently being incorporated into the Unix version of the Multipurpose Interactive Image Processing System (MIIPS) (Gudehus, 1989).

2. The FITS Library Package

2.1 Number of Axes

Although the FITS standard allows up to 999 axes, for image files the maximum of 3 dimensions, used in the FITS Library Package, serves most purposes. The number of axes is denoted by the keyword NAXIS, and the number of elements along axis n is given by NAXISn. Thus for a file representing a single row of pixels, NAXIS could be set to 1 and NAXIS1 set to the number of pixels. However, because images are two-dimensional and because extra planes could be used to represent, for exmaple, different wavelengths or real and imaginary parts, NAXIS is more likely to be 3. By default, when creating new files from scratch, the FITS Library Package uses NAXIS=3. If a file with a single row and a single plane is written, NAXIS2 and NAXIS3 are set to 1. However, If a file with NAXIS equal to 1 or 2 is read, and an operation performed on it, e.g. image arithmetic or cropping, and the output is written to the same file or to a new file, then the original NAXIS and NAXISn values are retained.

2.2 Examples

The FITS Library Package consists of source code for the Unix operating system and executables for a demonstration program compiled for Linux on

Intel processors, and Macintosh on the Power PC processor. The Macintosh Power PC program will also run on Macintoshes with the Intel processor. The demonstration program, testfits.exe, can carry out the following operations on one or more FITS files:

1. Reads all or part of a row of data
2. Plots a cut through a file
3. Reads an xy area
4. Displays an image on SAOimage (Enhanced version)
5. Writes all or part of a row for a new or existing file
6. Writes an xy area for a new or existing file
7. Performs simple rotations on a file
8. Performs simple arithmetic on a file
9. Performs a simple crop on a file
10. Copies one or more maps (image extensions) from one file to another
11. Copies a set of maps from one file to a set of planes in another file
12. Prints the header
13. Writes a COMMENT or HISTORY keycard to the header of an existing file
14. Creates or updates a keycard with a string value, e.g., AUTHOR, CTYPEn, BUNIT, DATE, DATE-OBS, etc.
15. Creates or updates the BSCALE and BZERO keycards for an existing file
16. Replaces a keycard with blanks

The demonstration program serves as an example of how to use the package subprograms to create custom applications. Files created or modified with the FITS library routines have been tested with the NASA FITS File Verifier program, accessible at http://fits.gsfc.nasa.gov/fits_verify.html. The subroutine openfits, called by each test example, opens a FITS file for reading or writing. For a new file, or a new map to be added to an existing file, the parameters describing the map, i.e., the number of columns, rows, and planes, and the data type, can be supplied in the call or can prompted for.

Test example 1 displays on a terminal window, the pixel values from all or part of a row of data in a map. The file is opened with a call to openfits and the data is read with a call to rdfitsrow or rddpfitsrow for double precision data.

Test example 2 makes use of the MIIPS Plot Package (Gudehus, 1988) to plot one or more rows or columns in one or more maps, or a single plot at a chosen angle. For the case of a plot at an arbitrary angle, the location of the center of the cut may be input manually or interactively with the enhanced version of SAOimage (Gudehus, 1995). The version of SAOimage which supports such input is v. 1.2E obtainable from http://www.chara.gsu.edu/~gudehus/enhanced_saoimage.html.

Test example 3 displays on a terminal window, the pixel values from an xy area of data in a map. After the file is opened, the data is read with a call to rdfitsxy or rddpfitsxy for double precision data.

Test example 4 displays an image on the enhanced version of SAOimage. Here a call to the subroutine fits_to_sao is made. This subroutine opens the file, collects display information from the user, sets up SAOimage, fills an array with calls to rdfitsrow or rddpfitsrow, and then loads the data with a call to saoload.

Test examples 5 and 6 are simple examples of writing pixel data to an old file, a new map in an old file, or a new file. Because there is no change in size of the map, the input unit number can be the same as the output number. For a new file, openfits will prompt for the number of maps, columns, rows, planes, and data format. Depending on whether a row or an xy area is to be written, calls to wrfitsrow or wrfitsxy, respectively, can be made.

Test example 7 is a simple example of rotating a file. One can select a group of maps within the input file that are to be rotated. All planes within each map are rotated by the same amount. This simple example has some limitations in that the output file cannot be the input file, and the headers of the input maps are not retained.

Test example 8 performs simple arithmetic on a file by adding a chosen constant (or bias) to all the pixel values in a single plane of a single map, and than multiplying by a chosen factor. This example permits the widest range of possibilities for saving the ouput. One can save to the original map and plane, save to the end of the same file as a new map, save to the end of a different existing file, and save to a new file. In each case the information in the original map header is retained. This is accomplished by creating a temporary file. To finish up, the user is prompted to Save, or Save As with a call to subroutine save_save_as.

Test example 9 performs simple cropping of 1 or more maps in a file. All the maps will be cropped to the same size and at the same starting coordinates. All planes in the maps are cropped as well. This example also permits saving to the original maps and planes, saving to the end of the same file as a sequence of new maps, saving to the end of a different existing file, and saving to a new file. In each case the information in the original map headers is retained. Again, a temporary file is created and the user is prompted to Save or Save As.

Test example 10 permits one to copy a selection of FITS maps from one file to another file. A call to copy_fits_maps first calls get_input_fits_maps in order to open the input file and prompt the user for the input maps. A call is then made to copy_fits_records which handles the details of copying of records from one file to another, taking into account how the the organization of the output file depends on the initial and final number of maps in the output file.

Test example 11 copies a selected set of maps from one file to a set of planes in a map in another file. This test demonstrates the high level routine fits_maps_to_planes. After opening the file and prompting for the input maps, the subroutine checks that all the maps have the same pixel dimensions. Then the data formats of the input maps are compared, and if they differ, they are displayed and the user is prompted to select an ouput data format. Usually one would select a format that does not sacrifice precision. Currently, this example only permits saving the results to a file other than the input file.

Test example 12 prints the header of a FITS file. It also counts the number of blank and nonblank keycards. This test makes use of the subroutine mhfitscrd. After opening the file, the user is prompted for the map number if there are more than one. If the file has one map, the single header is displayed on ther terminal window; if the file has more than one map, the principal header plus the selected header are displayed.

Test example 13 writes a COMMENT or HISTORY keycard to the header of an existing file and demonstrates the usage of subroutine wrfitsmhst. One can enter up to 72 characters, and duplicate/create or replace a keycard. Because multiple occurences of these two keycards are allowed, when replacing a keycard the user is prompted for which occurence of the keycard to overwrite.

Test example 14 creates or updates a keycard with a string value, e.g., AUTHOR, CTYPEn, BUNIT, DATE, DATE-OBS, etc. One can enter up to 20 characters for the value, and up to 47 characters for the comment. For

these keycards, the FITS standard allows only one instance of each to be used in a map, so the user is never prompted for which occurence to overwrite.

Test example 15 creates or updates the BSCALE and BZERO keycards for an existing file. Again, because for these keycards only one instance of each is allowed, the user is never prompted for which occurence to overwrite.

Test example 16 replaces a keycard with blanks. The subroutine wrfitsmhst will prompt the user if there is more than one instance of the keyword. This command can be useful if the header needs to be revised, or if it contains keywords that have become deprecated.

References

Gudehus, D. 1988, *The MIIPS Plotting Package,* Working Group on Astronomical Software, 172th AAS Meeting, Kansas City, MO, Bull. A.A.S. **20**, 708.

Gudehus, D. 1989, *MIIPS, An Image Processing System,* Working Group on Astronomical Software, 174th AAS Meeting, Ann Arbor, MI, Bull. A.A.S. **21**, 784.

Gudehus, D. 1995, *A Communications Interface for SAOimage,* Pub. A.S.P., **107**, 199.

Wells, D. C.; Greisen, E. W.; Harten, R. H. 1981, *FITS - A Flexible Image Transport System,* A & Ap Suppl. Ser. **44**, 363.