ASTR 288C – Lecture 6

Monday, 12 October 2009

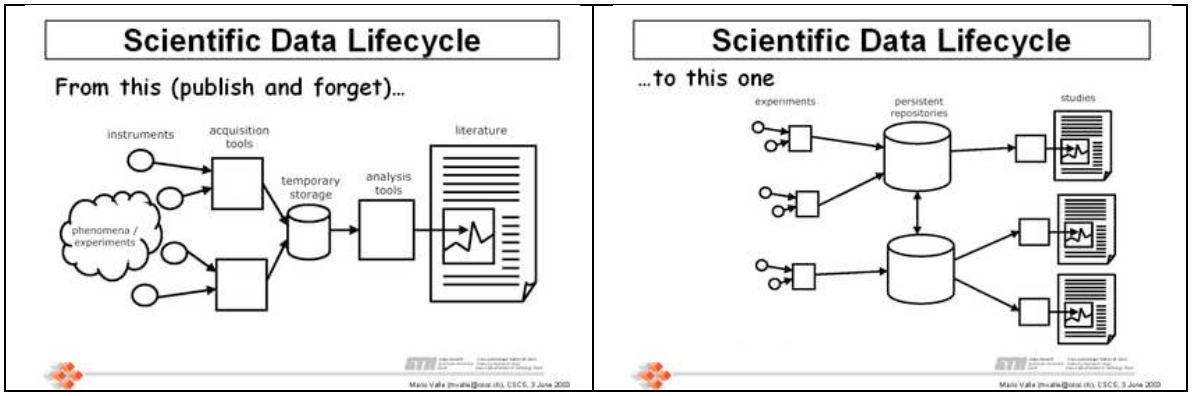
Scientific Data Format

Introduction

Data formats are a critical, and often overlooked part of the scientific process. There is little point in collecting data if it can not be used afterwards. Data needs to be saved in a form that allows it to be easily used, and retain all of the original information. In many cases data will be archived and may be used long after the original observations have been written up an published, so it is critical that data be stored in formats that allow researchers to use it in the future.

There are two underlying goals to scientific data formats. The first is to create a format that allows data to be stored in a way that is convenient, flexible, and portable. The second is to format data so that it can be stored in long-term archives that are easy to search and retrieve data from, and easy to use the data once it has been retrieved.

Historically the final resting place for scientific data has been in the scientific literature. Scientists publish tables of data in refereed journals or peer-reviewed books and other researchers use the data in these articles. The advantage of this system was that the data was fairly easy to find. It was located with the publication where it was originally presented. The disadvantage of this system is that it can be difficult to convert data that has been published in a table in an article into a form that is useful to other researchers. In general data had to be copied from an article by hand, which is slow, manpower intensive, and can easily lead to transcription errors. Another problem is that very large data sets, and raw data were usually not published and tended to remain in the hands of the original researcher. This often led to data being lost.



Modern data storage tends towards archiving. The emphasis is on publishing results in the refereed literature and describing the data. Samples of data are

usually published along with a pointer to the long-term archive that contains the totality of the data. This has the advantages that data can be searched and retrieved by the astrophysics community, and that large amounts of data can be archived for future use. Another advantage is that the data is automatically in an electronic form, which makes it easy to use.

The rapid growth of data archives since the 1990s has made it important to pick data formats carefully. To be useful a format needs to have the following qualities.

- Portable
  Data needs to be stored in a way that allows it to be shared between scientists. This means that it needs to be accessible on any computer system and not use proprietary data formats, such as GIF.

- Lossless
  Data must be stored in a way that no information is lost in the process of storing or recovering the data. This means that lossy data formats such as JPEG are not acceptable as a way of storing scientific data. Any compression that is done on data must be done in a way that preserves all of the information in the data. This is particularly true for raw data. High level data products that have been produced by analysing or processing the raw data can be truncated to the precision of the data *if* the raw data is available to allow the high level products to be recreated in the future. This is acceptable because the precision of calculations is often much higher than the precision of the data. The digits beyond the precision of the data do not carry useful information, so they may be removed to save space. However, as disk space becomes cheaper there is less need to do this sort of data truncation.

- Indexable
  Data must be stored in a way that allows it to be indexed and searched. Ideally, this is done by storing data using self-describing data formats. For example, each data file includes a description of what is in the file.

ASCII (or Plain Text)

The American Standard Code for Information Exchange (ASCII) is the de facto standard for representing characters on computers. It is widely used to store information in plain text files. An example of an ASCII file is the following list of distances from short–hard gamma-ray bursts to the centres of their host galaxies.

```
# Offsets of short-hard GRBs from their putative host galaxies.
#
# Sources: 1 - Berger et al. (2005, Nature, 438, 988)
#          2 - Fox et al. (2005, Nature, 437, 845)
#          3 - Soderberg et al. (2006, astro-ph/0601455)
#          4 - astro-ph/0901.4038v2
# GRB    kpc    r/r_e   Source
#
 050509  44      13       2
 050709   3.8     1.8     2
 050724   2.57    0.4     1
 051221   0.760   0.29    3
```

The file contains metadata describing what is in the file, and four columns of numeric data.

Plain text files are widely used for quick-and-dirty data storage, but not widely used for raw data, or for archive-quality final data. The advantages of plain text are that it is easy to create ASCII files and easy to edit them. This means that it is generally straightforward to create ASCII files in any given format, and meta data can be added or edited as needed. Plain text files can be read by most data manipulation packages, and can usually be understood by a human. Plain text data files tend to be useful for the intermediate stages of data analysis, such as input for plotting routines, or isolating a subset of the data for a specific data analysis. Plain text files are also useful if you want to experiment with a small subset of data to see how certain changes affect the results of an analysis.

A big disadvantage of plain text files is that they are not inherently self-documenting. Any metadata must be deliberately written to the file, and there is not standard way of presenting the meta data. In the above example the "r/r_e" column is not described. There is no way of knowing, from the contents of the file, what the numbers in this column represent (in this example it is the ratio of the distance of the GRB from the centre of the host, r, to the effective radius of the host, r_e). Similarly, there is no description of the "kpc" column. One can not tell what it represents from the contents of the file alone (in this example it is the distance of the GRB from the centre of the host in kpc). .

An even more fundamental problem with using plain text files to store astronomical data is that the structure of the file is not specified within the file. In this case it is fairly easy for a human to see that there are four columns of data. However, in the following example it is not clear if the data is four columns, or a 4 × 4 matrix.

```
 1.012   2.111  2.111  0.000
 2.909   8.321  4.534  0.000
 5.132   9.876  1.357  0.000
 0.000   0.000  0.000  0.000
```

Finally, plain text data files, while very efficient for dealing with small amounts of data, they rapidly become very large when dealing with large amounts of data, such as images. Binary representations of such data can take significantly less space. This is particularly true of raw data sets, which can be very large.

XML

XML, or Extensible Markup Language, is not a true data format. It is a way of adding meta-data to ASCII files so that the file becomes self-documenting. If an XML file is constructed correctly the file can stand alone and tell any user (or software) exactly what is in the file and what the structure of the data is. XML is a generalized form of HTML that was designed to transport and store data with the goal of describing what the data is. An example of XML is given here.

```
<photometry>
 <photometry source="IMC930475">
  <date>2008-04-14</date>
  <starttime>03:12:15</starttime>
  <exposure>2000</exposure>
  <filter>R</filter>
  <magnitude>15.87</magnitude>
  <magerror>0.03</magerror>
 </source>
 <photometry source="IMC930476">
  <date>2007-02-28</date>
  <starttime>11:42:04</starttime>
  <exposure>2000</exposure>
  <filter>R</filter>
  <magnitude>18.23</magnitude>
  <magerror>0.15</magerror>
 </source>
</photometry>
```

In this example every entry in the file is tagged with a descriptor that says what that entry is. In fact, there is more meta data in this example that there is actual data. Software can read this meta data description and deal with the actual data in an appropriate way. In principle any data can be stored in XML files, but they suffer from the same size problem as simple ASCII files do–they rapidly become very large, and thus are not well suited for image data or large quantities of raw data. It is not unusual for the majority of an XML file to be meta data. The XML format can be well suited for reduced data.

The advantages of XML are similar to the advantages of plain text, except that XML files are self-documenting. The data is described within the file, and the structure of the file is described by standardized metadata. The disadvantage is that XML files are not easy for a human to read. They generally require software to present the data in a way that a human can understand it. Other disadvantages are that XML is not always suited to very large data sets, and the large amount of

metadata needed can result in very large files.

Flexible Image Transport System (FITS)

Description

The Flexible Image Transport System (FITS) is a file structure that was developed to be a standard way of storing astronomical data, such as images and spectra. Prior to the late 1980s almost every observatory stored its data in a different format. These formats were often optimized to a particular instrument or a particular computer system. Several attempts were made to develop standard data storage formats for astronomical data, but FITS has been the most successful.

FITS was originally developed in the late 1970's as an archive and interchange format for astronomical data files. Since the 1990s FITS has also come into wide use as an on-line file format that can be directly read and written by data analysis software. FITS is much more than just another image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multidimensional arrays and 2-dimensional tables containing rows and columns of data. Data that is stored in a FITS file is self-documenting.

A FITS file consists of one or more Header + Data Units (HDUs), where the first HDU is called the `Primary HDU', or `Primary Array'. The primary array contains an *N*-dimensional array of pixels, such as a 1-D spectrum, a 2-D image, or a 3-D data cube. Five different primary data types are supported: unsigned 8-bit bytes, 16 and 32-bit signed integers, and 32 and 64-bit single or double precision floating point reals. FITS can also store 16 and 32-bit unsigned integers.
Any number of additional HDUs may follow the primary array; these additional HDUs are called FITS `extensions'. There are currently 3 types of extensions defined by the FITS Standard (defined at http://archive.stsci.edu/fits/fits_standard/).

- Image Extension
  An image extension is an *N*-dimensional array of pixels in binary format, like in a primary array.
- ASCII Table Extension
  This is an extension that contains rows and columns of data in ASCII character format.
- Binary Table Extension
  This is an extension that contains rows and columns of data in binary representation.

Every HDU consists of an ASCII format `Header Unit' followed by an optional `Data Unit'. For historical reasons, each header or data unit must be an exact multiple of 2880 bytes long. Any unused space at the end of the header or data unit is padded with fill characters (ASCII blanks or NULs depending on the type of unit).

Each header unit consists of any number of 80-character keyword records, which have the general form:

KEYNAME = value / comment string

The keyword names may be up to 8 characters long and can only contain uppercase letters, the digits 0–9, the hyphen, and the underscore character. The keyword name is (usually) followed by an equals sign and a space character (=) in columns 9 and 10 of the record, followed by the value of the keyword which may be either an integer, a floating point number, a character string (enclosed in single quotes), or a boolean value (the letter T or F).

The last keyword in the header is always the `END' keyword, which has no value or comment fields. There are many rules governing the exact format of a keyword record (see the FITS Standard for details) so it is generally better to rely on standard interface software like CFITSIO or FTOOLS to correctly construct or parse the keyword records rather than directly reading or writing the raw FITS file.

Each header unit begins with a series of required keywords that specify the size and format of the following data unit. A 2-dimensional image primary array header, for example, begins with the following keywords:

```
SIMPLE =                    T / file does conform to FITS standard
BITPIX =                   16 / number of bits per data pixel
NAXIS  =                    2 / number of data axes
NAXIS1 =                  440 / length of data axis 1
NAXIS2 =                  300 / length of data axis 2
```

The required keywords may be followed by other optional keywords to describe various aspects of the data, such as the date and time of the observation. Other COMMENT or HISTORY keywords are also frequently added to further document the contents of the data file.

The data unit, if present, immediately follows the last 2880-byte block in the header unit. Note that some HDUs do not have a data unit and only consist of the header unit.

An example of the structure of a FITS file is given below.

- ftlist example.fits H

```
Name      Type              Dimensions
----      ----              ----------
HDU 1     Primary Array     Null Array
HDU 2     wh250005527I      Image          Real4(1138x1141)
HDU 3     MAGHIST           BinTable       69 cols x 5 rows
```

In this example the Primary Array (HDU 1) is empty, except for a set of keywords that describe the data in the file. The first extension (HDU 2) is an image with

dimensions 1138 × 1141 and the data is stored as 4-byte real numbers. Notice that there are different types of extensions in a single file. This file contains a null array (HDU 1), two-dimension image (HDU 2), and a binary table (HDU 3). Each image extension is independent of the others in the file.

There is a lot of information about FITS available at http://swift.gsfc.nasa.gov/docs/heasarc/fits.html . Although FITS was originally designed to be a standard format for astronomical data there are several different versions of FITS. The differences are mostly minor and generally consist of different versions of FITS using different keyword names. For the most part all versions of FITS files can be read by the major software packages.

Software

FITS files are not intended to be read or modified directly by humans. They are intended to be used in conjunction with software that reads, writes, and modifies FITS files. The standard software package for working with FITS files is the FTOOLS software developed at NASA's High Energy Astrophysics Science Archive Research Centre (HEASARC; http://heasarc.gsfc.nasa.gov/). FTOOLS can be obtained from http://swift.gsfc.nasa.gov/docs/software/lheasoft/download.html . There is also a set of subroutine libraries, called FITSIO, that can be used to work with FITS file in various programming languages. FITSIO is described in detail at http://swift.gsfc.nasa.gov/docs/software/fitsio/fitsio.html . Supported programming languages are

- C++
- C#
- Perl
- Tcl
- Python
- Ruby
- S-lang
- MatLab
- LabVIEW
- Photoshop

ftlist

FTLIST is an FTOOL that prints out the contents of a FITS file. For a detailed description of FTLIST see the help page for FTLIST (type fhelp ftlist) at the command line prompt. FTLIST has five modes, each of which returns a different type of information about the FITS file. The modes are

- H—prints a one-line summary of the contents of each HDU in the FITS file. This is good for determining the general structure of the file, and for identifying which HDU the data you are interested in is in.
- C—prints information (name, units, etc.) about each column in each FITS table. This works for ASCII and binary table HDUs.

- K—prints the keywords and their values for each HDU. This works for every type of HDU.
- I—prints the pixel values of the image array. This works for image HDUs.
- T—prints the contents of a FITS table. This works for ASCII and binary table HDUs.

More than one mode can be specified at a time. For example, to print out the HDU summary (H option) and the header keywords (K option) in the first extension (HDU 2) of the input file "infile.fits" and exclude the COMMENT keywords type

- `ftlist infile.fits+1 hk exclude=comment`

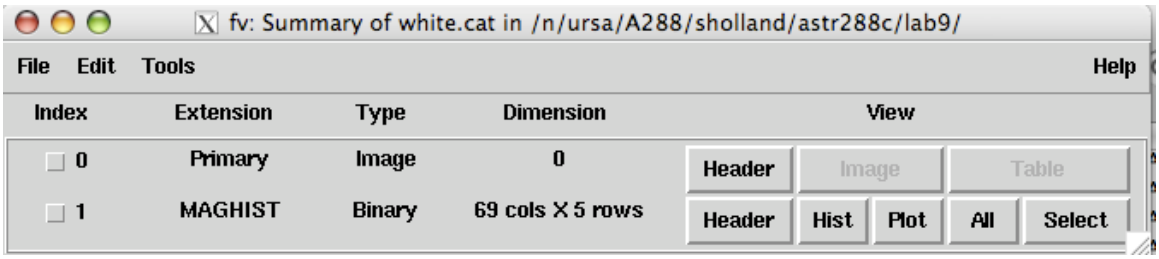
FITS files can contain a large amount of information. It is not unusual to have a FITS file with several dozen columns of data. Because of this is often useful to view only a subset of the data in the file. The help page for FTLIST contains detailed information about how to do this.

fv

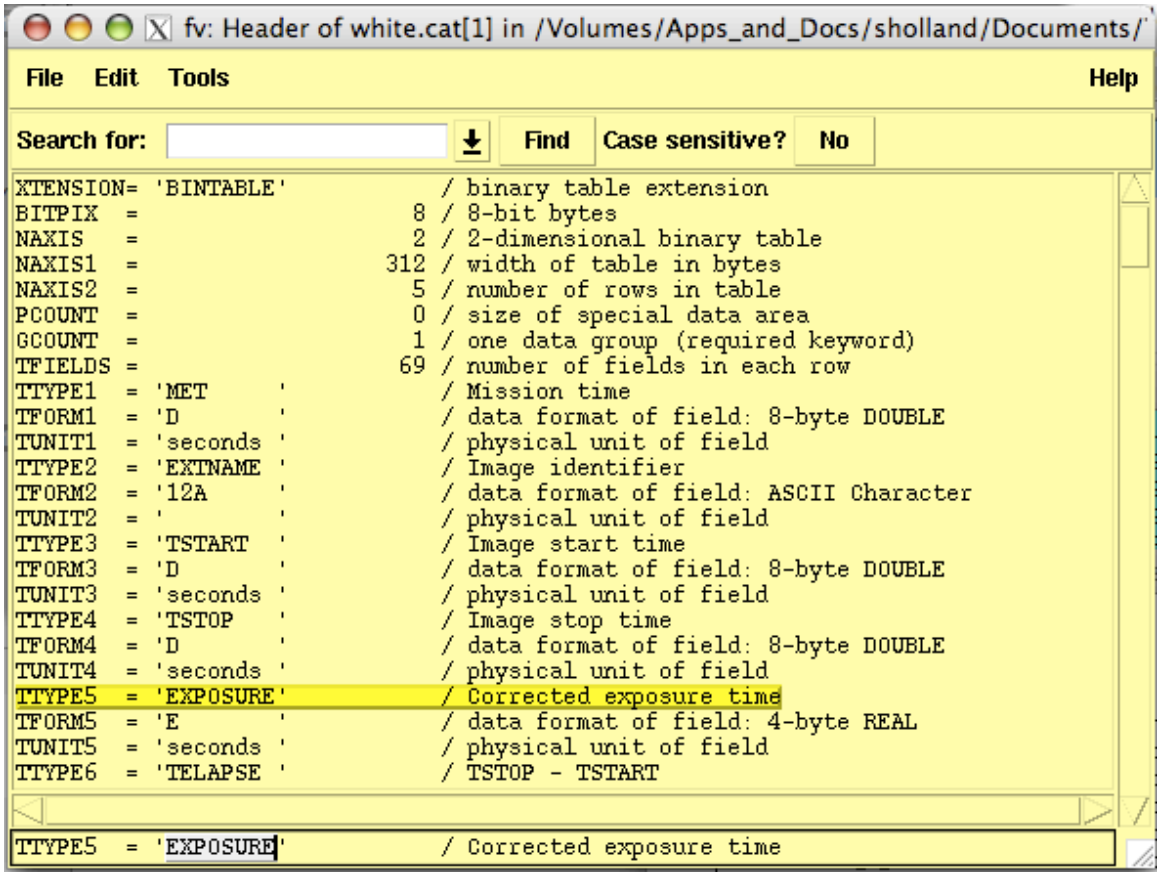
FV is an interactive browser, editor, and viewer for FITS files. To start FV simply type fv at the command line. This will put up a window that looks something like this.

This is the main menu for FV. From this one can create new FITS files, open an existing file, access various catalogues and databases, and run FTools on your FITS file.

When a FITS file is open a dialogue box that looks a bit like this will appear.

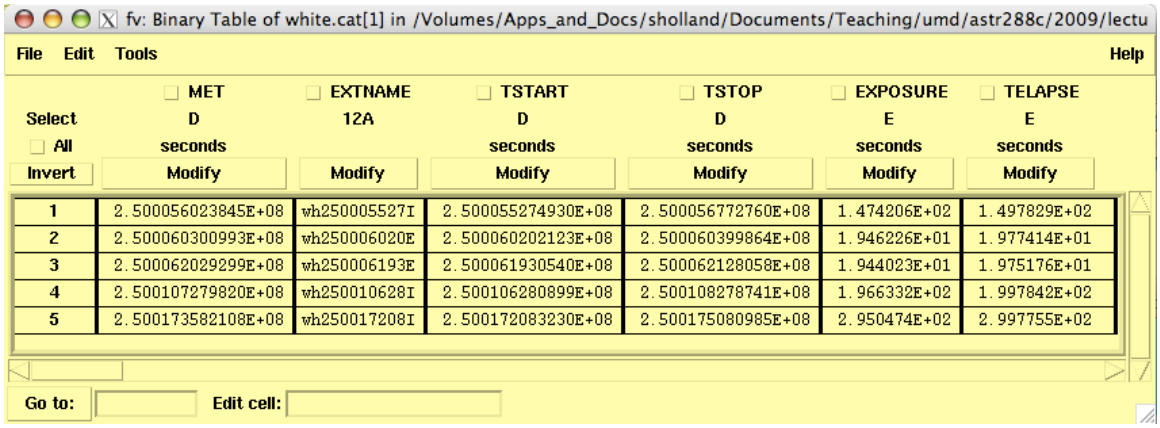


The "Header" button opens a window that allows one to view and edit the keywords for the chosen extension. For example, the header of the MAGHIST extension looks like this.



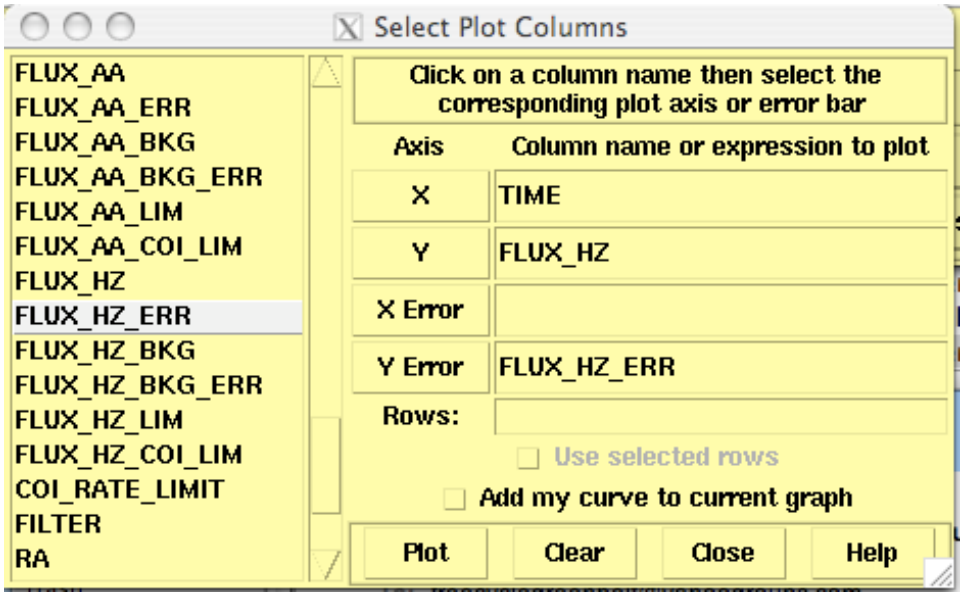
Notice that editing is done in the one-line box at the bottom of the window. To edit a keyword select the line that you are interested in and edit it when it appears in the bottom line. Edits do not take affect until the <return> key is pressed. Changes to a file need to be saved by selecting "save" under the "file" menu.

Selecting the “All” button lets one view the data in the FITS extension. The data in the MAGHIST extension looks like this.



To edit a cell select the cell and do the editing in the “Edit cell:” box at the bottom of the window. Press <return> for the edit to take effect. Be sure to save changes using the “File” menu.

Data can be plotted by selecting “Plot”.



Any column can be plotted on any axis, or as an error bar, by selecting the column of interest and clicking on the axis of interest. Alternately, just type the name of the column directly.

Images can be displayed and manipulated in similar ways.

Lab Work

The goal of this lab is to gain some familiarity working with FITS files. To do this we will examine two FITS files. One that contains a *Swift*/UVOT image of the optical afterglow of GRB 090812, and one that contains a light curve for the optical afterglow of this gamma-ray burst. We will also use FV to create a new FITS file.

First, find the following files that you used last week: "sw00359711000uwh_sk.img" and "sw00359711000uwh_sk.cat". If you do not have these files then you can download them from http://lheawww.gsfc.nasa.gov/~sholland/astr288c/autumn_2009/index.html . Move this file to your working directory.

Viewing FITS Files

FV is a very powerful tool for interactively examining FITS files.

1. Log into your account, download the two files, and put them into your working directory.

2. Start FV.

   - `astroload heasoft`
   - `fv &`

3. Open "sw00359711000uwh_sk.cat" and make a plot of the flux density of the source, FLUX_AA (with error bars from FLUX_AA_ERR) as a function of TIME. Print this plot and hand it in as part of your homework. You can close the plotting windows now if you wish.

4. Choose "All" for the MAGHIST extension in the summary window to answer this question. The MAGHIST extension contains one row for each image in the "sw00359711000uwh_sk.img" file. Look at the PLATE_SCALE column in the MAGHIST extension. Is the plate scale the same for all five of the images? What is the plate scale of the last image? Hand this in as part of your homework. Include the units. You can close this window if you wish.

5. Close the "sw00359711000uwh_sk.cat" window and open the file "sw00359711000uwh_sk.img".

6. What is the value of the DATE-OBS keyword for the last extension in this file? Hand this in as part of your homework. Use the "Header" button.

7. What is the value of this image (extension wh271749819I) at row 230 column 322? Hand this in as part of your homework. Use the "Table" button.

Creating a FITS File

There are several ways to use FTools to create a FITS file. One way to create a FITS file that contains a simple image is to use FV. The procedure is as follows.

1. Start FV

   - fv &

2. Select "New File…". This will open a create image box. Enter a filename. This can be anything that you want, but it should end with ".fits" so that it can be easily identified as a FITS file. One possibility is YOURNAME.fits. Let's assume that the file is called "fred.fits". Set the image data type to be "Float (−32)". Set the image dimensions to "8,8" (without the quotation marks). This indicates that the image should have 8 rows and 8 columns. Now, click on "Create".

3. Add the following keyword to the header data. Do this by clicking on "Header" in the summary window that pops up at the end of step 3. Select the "END" keyword then choose "Insert Key" from the Edit menu. Now, add this keyword.

   CREATOR = 'YOUR NAME' / Person who created this file

   The single quotes must be included. Replace YOUR NAME with your name. Save the changes from the File menu.

5. Set the image values by clicking on "Table" in the summary window. Set the pixel values to anything that you like. You may leave some of them as zero, but not all of them. Edit a pixel value by selecting a cell (pixel) and typing the new value in the "Edit cell:" box. Press <return> to enter the new value. When you are done save the image from the File menu. You can view your image with the "Image" button in the summary window.

6. Create a binary table extension using the "New Extension…" item in the Edit menu.

7. Add a column of data to the binary table extension by pressing the "All" button for that extension then choosing "Edit"→"Insert"→"Column". The new column should have the following properties.
   Column Name: *Data*
   Column Format: choose 4-byte real from the menu next to the format cell.
   Column Unit: leave blank
   Display Format: leave blank
   Insert Before: End of Table

8. Edit the value in row 1 column 1 so that it is 1.12358 and then save the changes.

9. E-mail your final FITS file to Stephen.T.Holland@nasa.gov . This will be part of your homework for this week.