Eclipse User's Guide

Nicolas Devillard

March 11, 2002

Contents

1 Introduction

eclipse is a library written in ANSI C, offering services dedicated to astronomical data processing. There are services to read/write FITS files, to perform image processing, 3d filtering, compute photometry, image quality, statistics, manipulate image files, etc. These services are all accessible to the C programmer directly, the documentation is contained within the source code in the header files. If you want to use **eclipse** as a base for your software development, it is recommended to read the *eclipse developer's manual*, available from the **eclipse** home page. Notice that outside developments using the **eclipse** library are not supported by ESO as such, you are on your own.

Most **eclipse** users however, are not programmers. They use the services offered by the library through Unix commands provided in the Web distribution of this software. The number of Unix commands and their roles have been changing over time. Some commands have merged, some have been split into several commands, the software has evolved quite a lot since its beginning in 1995.

The good thing about the **eclipse** Unix commands, is that they are very lightweighted. To divide an image by another, no need to startup a data reduction engine, convert the images to a local format or whatever. In a single command-line you can issue the command and get the result back in FITS format too, ready to interact with other image files. The learning curve is the same as for Unix commands, i.e. see it once and know how to find the man page, that is all you need.

Unix commands are also conveniently used for scripting in readily any scripting language. You can build complex data processing programs by scripting **eclipse** commands from the Unix shell, Python, Tcl/Tk, Perl,... any virtually kind of program that can launch Unix commands.

It is generally a good idea to do so for prototyping phases of a data reduction project, for one-shot tasks, and in general every time you do not want to get into programming, but just want the data reduced. Programming a scripting language is learned quickly and developped at lightspeed. However, it is a very bad idea to plan on scripts to perform heavy data reduction on a daily basis. The reason is simple: if you have to call at least 2 Unix commands to perform your processing, you need to send the data back to the disk between the 2 command calls, because the inputs to Unix commands are *file names*. If you have lots of data, that translates into heavy I/O which becomes then an obvious bottleneck for your process. For such cases, you would certainly prefer to use the services as offered at the C level and program your processes in C directly.

For example, the Adonis data reduction does not handle enough data to flood a disk and a CPU. To reduce one night of data, a good workstation needs no more than a few hours to apply the basic calibrations. Making scripts for that purpose is ideal: they are developped quickly and since efficiency is not at stake, they do the job quite nicely.

On the other hand, ISAAC/SOFI are dealing with batches of data reaching a gigabyte in size, which means that efficient memory handling becomes a necessity, and avoiding disk I/O is crucial. Such efficiency can only be reached by dealing with the data at the C level, thus a list of Unix commands dedicated for ISAAC/SOFI data reduction only. These commands are called *recipes* in the DataFlow naming scheme.

Last, **eclipse** offers some services on the command-line for data reduction, and some commands do basic data analysis. But this software is in no way a complete data analysis environment as MIDAS, IRAF, or IDL offer. If you are looking for an environment to interact with your data, you will be disappointed by the little (if any) interactions offered by **eclipse** commands. This library is aimed at pipeline data processing, i.e. number crunching following a list of user-defined parameters, without user interaction. **eclipse** is a data reduction engine, not a data analysis facility.

2 Download and installation

eclipse can be downloaded from the following WWW site:

`http://www.eso.org/eclipse/`

The usual way is to download the source code and compile it, following the instructions given both on the Web pages and in the distributed tar files. If you are using Linux on an i386 platform, you can also download the associated RPM file containing both binaries and documentation. Installation is much simpler then.

3 File conversion

The commands in this section allow to convert an image to PPM format, mostly for compatibility with other image processing packages.

3.1 3color

`3color` takes in input 3 compatible image files and produces a 24-bit color image in PPM format. This is useful to build colour composites out of images taken in different observation bands. Notice that the input images must be compatible, i.e. have the same size and be aligned, in order to get a sensible result.

3.2 fitsppm

`fitsppm` converts a FITS file to 8-bit pixels, then to PPM format. Dynamic range is lost in the process. This is useful to build image files to exchange data with other image processing commands.

4 FITS manipulations

eclipse supports only one image format: FITS. Although including support for other image file formats would be a minor task, it is not foreseen to do so for the moment. **eclipse** is about number crunching for astronomy, not PhotoShop.

FITS images have uncompressed pixel data following a header expressed in pure ASCII. The following commands allow searching, parsing, printing of these values.

4.1 dfits and fitsort

The twins of FITS header display, `dfits` and `fitsort` are very handy to get an overview of what is contained in large amounts of FITS files. `dfits` dumps FITS headers to its stdout, nothing more. The good side is that it accepts jokers on the command-line, allowing constructs such as:

```
% dfits *.fits | grep OBJECT
```

Getting a FITS header on a standard Unix stream allows to send this ASCII data to any kind of text-processing Unix command for further use. See the standard suite of Unix text tools such as `cat, more, sed, ed, awk, join, ...` and languages such as Perl or Python.

Notice that `dfits` knows how to display the main header of a FITS file, but also extensions (see the -x option).

`fitsort` works only by accepting data from `dfits`. They are meant to be used chained in a Unix pipe. With `fitsort`, you specify the names of the keywords you want to see in output. It will pick up these keys for all the files it can find on its stdin, sort them out, and print them in a readable fashion on a terminal. You can also re-direct the output to an ASCII file and read it with any kind of spreadsheet program. `dfits` and `fitsort` are used daily in many observatories for simple display over large numbers of FITS files.

4.2 dtfits

Most **eclipse** commands are written to deal with images or cubes in FITS format, but the need to support FITS tables has arisen with some of the latest instrument recipes. Some ISAAC recipes expect FITS table in input, or produce FITS-compliant tables in output. This means that most users dealing with these tables need a way of dumping them to see what they contain, with no more than a command-line.

`dtfits` offers this service: it accepts a single name on its command-line, and will dump header information and contents of the given FITS table (be it ASCII or binary).

4.3 fitsedit

If you are looking for software being able to edit/modify FITS headers, the `fitsedit` command is not supported any more under **eclipse** . You may want to have a look at FTU, another ESO-supported tool, downloadable from:

`http://archive.eso.org/ftu`

4.4 fitsmd5

This command computes the MD5 signature of all data parts contained in a given FITS file (including all extension data if present) and prints out the value on stdout. It can also update the main FITS header of a file if it already contains a keyword called `DATAMD5`.

This feature is meant to give a unique ID to a FITS file. It is a good solution to uniquely identify frames among a large list of FITS files that can be renamed or which FITS header can be modified. Most users will not need it at all, but people dealing with very large numbers of FITS files might want to evaluate it to solve frame identity issues.

4.5 hierarch28

The famous ESO hierachical keywords are unfortunately not compatible with many FITS readers (including the one in IRAF at the date of writing). If you badly need a software to read information from a file FITS header but the information is located in an `HIERARCH ESO` keyword, you want to use `hierarch28`. To use it, create a simple correspondance table translating ESO keywords to your own, then apply the command. Since this command works locally on the file, it does not need to load it into memory, working extremely fast.

4.6 iofits

FITS images may contain pixels in one of the following pixel formats: 8-bit integers, 16-bit integers, 32-bit integers, 32-bit floating point format, or 64-bit double precision floating point format. Converting image files from one format to another is not always an easy task, and you most probably would like to apply the same process over large number of files, e.g. to reduce the disk space they use.

`iofits` converts pixels from one depth to another. Be aware though that most conversions will incur some data loss if you are not careful about the dynamic range of the input images.

4.7 replacekey

To add information into a FITS file header, it is cumbersome to load it, modify the contents in memory, then dump the results as it is currently done by `fitsedit`. One way to avoid that is to modify the file in place, but that assumes that you will use some existing space in the file to insert your data.

`replacekey` is meant to replace some cards in a FITS header by a card you provide on the command-line. It acts in place, making it extremely fast. The ideal is to use a software that will put placeholders in the output files it creates. You can then insert information in place of the placeholders, by declaring to `replacekey` what the placeholder name is, and what you want to replace it with.

5 Image and cube manipulation

5.1 average

This command is meant to reduce 3d FITS files to a single 2d image. It offers several algorithms to filter the data in the 3d cube.

5.2 catcube

This is the complement to `average`. This command takes in input a list of images or cubes of the same size (in X and Y) and concatenates them all to a single 3d FITS file.

5.3 ccube

`ccube` is a cube computer. It accepts arithmetic expressions mixing numbers and FITS files to perform basic arithmetic. Do not search any further if you are looking for a command-line way to divide an image by a constant (normalization), or divide all planes in a cube by an image (flatfielding), or subtract an image from all planes of a cube (sky-subtraction), or compute the average image from 3 planes, etc.

5.4 dumppix

`dumppix` outputs pixel information from a FITS image to stdout, in ASCII format. This is convenient to examine closely image values. You can also use Perl or awk to filter the output to compute basic statistics over lines of pixels.

5.5 extract

`extract` is a multi-purpose extraction routine that can extract pixel data from FITS files in a variety of ways: extract part of an image from an image, image planes from a cube (with a given extraction rule), quadrants from an image or a cube, etc.

5.6 collapse

`collapse` is useful for spectroscopic images, when a signal is all distributed over the horizontal or vertical direction of an image. It sums up all pixels over a given direction, possibly rejecting some mad pixels according to some rule, and produces a 1d signal in ASCII format on stdout.

5.7 imgen

`imgen` is an image generator, handy to produce default images. If you are looking for complex image generation, data analysis environments usually have very complex ways of creating signals. The supported patterns so far are:

- 2d Airy function
- 2d Gauss function
- 2d Lorentz function
- Optical transfer function
- Point-spread function
- Random with uniform distribution
- Random with gaussian distribution
- Random with lorentzian distribution
- Binary disk
- Binary rectangle
- Uniform (constant) image
- 2nd degree polynomial in 2d
- Jitter test cube
- Mandelbrot set

5.8 ipaste

`ipaste` allows to "paste" an image into another, i.e. to replace the pixels in an image by pixels in another image.

6 Image processing

6.1 deadpix

`deadpix` offers two services: bad pixel detection (according to two possible methods), and bad pixel cleaning in which pixels are replaced by an average of their closest neighbours.

The two available methods for bad pixel detection are detailed in the manual page for this command.

Notice that both methods are largely outdated. There are many ways of detecting bad pixels for a given detector, both are now efficient for many reasons. If you want to get a reliable bad pixel map, best is to use a twilight flat-field procedure.

6.2 fft

`fft` is a simple 2d FFT on images. In forward transform, it takes single images in input and produces a cube containing two images in output: either (real,imaginary) or (amplitude,phase). The reverse transform works exactly the other way round.

For the moment, `fft` only supports images which size is a power of 2. If anyone has a request for other image sizes in astronomy, drop a mail to the authors. There are FFT libraries covering readily any kind of signal size, but no one has ever requested such a functionality. Furthermore, there is an FFT command in virtually any data analysis environment so this command is really there for completeness.

6.3 filt

`filt` supports a number of 2d image processing filters, including some user-defined kernels. The most useful filters are probably the median and contrast enhancement filters which clean out images when intended for e.g. press releases, and the contour detector for some extended objects.

6.4 flat

`flat` takes in input a list of files taken at twilight. It deduces the flat-field of the detector by observing this rapidly increasing or decreasing signal. Since it computes a characteristic curve per pixel, it can also easily detect bad pixels. This is actually a much better method to use for bad pixel detection, than the ones offered by `deadpix`, but it requires a specific data acquisition at twilight, which is not always easy or simply possible.

6.5 histog

`histog` computes an image histogram and prints out the results in ASCII on stdout. It is then easy to display the histogram with any plotting tool such as gnuplot. `histog` also offers a gnuplot output if this software is already installed on your system.

6.6 norm

`norm` allows to normalize an image or a cube with respect to their average, flux, absolute flux. It can also normalize all pixel values to the interval in [0..1], or scale all fluxes to a given value.

6.7 setpix

`setpix` takes in input an image or a cube, and an ASCII file describing pixel positions and values. It will replace the pixels in the file by the ones given in the description.

6.8 thresh

`thresh` applies simple thresholding to an image or a cube. It also offers the possibility to threshold an image to a binary map.

6.9 warping

`warping` is an image resampling utility that allows to apply a deformation to an image. The supported deformations are:

- Translation
- Rotation
- Scaling
- Linear transform
- Polynomial transform of any degree

It is useful to apply e.g. small rotations, translations, or scale differences to images to make them fit for 3-colour combination. It can also be used to correct images for detector deformations, usually modelled by a second-degree polynomial. `warping` uses high quality resampling kernels to perform the resampling, in a reverse warping fashion. To get more information about resampling in image processing, the reader is referred to the image processing litterature.

7 Statistics and analysis

Performing data analysis is not in the main job definition of **eclipse** . However, statistics extraction are an integral part of the library, since they are vital to many automatic algorithms. The Unix commands in this section offer access to these services, so that it might be useful to get this kind of information in a single command-line request.

7.1 encircl

`encircl` computes the encircled energy around a specified position in an image. This command should be merged with another object analysis command in the future.

7.2 peak

`peak` is a general command for object detection and statistics computation over each object in a given image. It supersedes the previous `detpeak`, `fwhm`, `photfits` and `findobjs` commands since it can produce the same output as these commands did. `peak` is a fairly complete tool with a complex command line.

7.3 stcube

`stcube` is a very fast command that allows to perform basic statistics computation over many images, in a single command-line order: `stcube *.fits`.

`stcube` will return the min, max, mean, standard deviation, energy and absolute flux for all planes of all files given on the command-line. It works extremely fast, also for big images.

The median finder (`stcube -m`) is also very efficient, and useful to get an estimate of the background value in images of e.g. crowded fields.

7.4 strehl

This parameter is mostly of interest for people working in the Adaptive Optics domain. It has been customized for Adonis and is recommended to use with great caution in the parameters you provide in input.

The Strehl ratio is a criterion indicating the quality of an Adaptive Optics correction. See the litterature in this domain for a more complete definition, or the `strehl` manual page.

8 Miscellaneous

8.1 poisson

This stand-alone program generates "random" jitter offsets in a box with a Poisson law so that points are optimally scattered within the box. This program is included in ISAAC and SOFI main control software for the AutoJitter templates. It might be useful for users who want to simulate an observation with these instruments.

8.2 speedtest

This program tries to benchmark your machine for image processing tasks with **eclipse** and gives you an indication of how much faster (or slower) than an UltraSparc it is. Pretty useless stuff, actually.

8.3 std

eclipse contains hardcoded in its code a list of standard infrared stars used by VLT astronomers to calibrate ISAAC for zero-points.

The `std` command is meant to give access to this (small) catalog of stars, to search e.g. for the closest star from a position, with a magnitude above say 11 in band Ks. It offers several search criteria and result display styles. Do not expect a database server with SQL requests, it is only there to provide information about the stars contained in the code.

9 Instrument specific commands

The **eclipse** library has been used as a basis for the development of a number of pipelines. At the time of writing (July 2001), the following are supported by ESO:

- Adonis (maintained but not extended).
- ISAAC (maintained and further developped).
- CONICA (still experimental).
- WFI (still experimental).
- CCD (still experimental).

10 Scripting eclipse

If you want to write pipelines, you probably need efficient and specialized programs to perform processing tasks. This calls for C development. However this is often a big effort, probably too much if you are considering chains of very simple operations to be automated. One solution is to script **eclipse** from a high-level scripting language, such as Python, Perl, Tcl, or Lua.

eclipse is distributed with two sets of bindings: for Python and Lua. The following sections try to describe in a little more details what is actually available and what is meant by *scripting*.

10.1 Scripting a C library

A C library is a list of functions offered to a C programmers as a list of services, together with an associated.

10.2 Little pipelines in eclipse-lua