Astronomical Data Analysis Software and Systems X
ASP Conference Series, Vol. 238, 2001
F. R. Harnden Jr., F. A. Primini, and H. E. Payne, eds.

ESO C Library for an Image Processing Software Environment (eclipse)

Nicolas Devillard

European Southern Observatory

Abstract. Written in ANSI C, eclipse is a library offering numerous services related to astronomical image processing: FITS data access, various image and cube loading methods, binary image handling and filtering (including convolution and morphological filters), 2-D cross-correlation, connected components, cube and image arithmetic, dead pixel detection and correction, object detection, data extraction, flat-fielding with robust fit, image generation, statistics, photometry, image-space resampling, image combination, and cube stacking. It also contains support for mathematical tools like random number generation, FFT, curve fitting, matrices, fast median computation, and point-pattern matching. The main feature of this library is its ability to handle large amounts of input data (up to 2 GB in the current version) regardless of the amount of memory and swap available on the local machine. Another feature is the very high speed allowed by optimized C, making it an ideal base tool for programming efficient number-crunching applications, e.g., on parallel (Beowulf) systems.

Running on all Unix-like platforms, eclipse is portable. A high-level interface to Python is foreseen that would allow programmers to prototype their applications much faster than through C programs.

1. Introduction

The eclipse library is intended as a low-level software facility to offer image-processing and data analysis functionality through calls to a C library. End-users can also access most functions through Unix commands, but the main idea is to program stand-alone pipeline commands running without user interactions.

Begun in 1995 to provide Adonis users with a usable pipeline environment to reduce their data in real-time in the telescope control-room, eclipse has been further extended to support more instruments at ESO (ISAAC, CONICA, WFI, possibly VIRMOS) and also in other institutes (Subaru, CFH).

2. Base requirements

From the eclipse developer's manual, its base requirements include:

- It shall be freely distributed through the Net.
- It shall be easily compiled on most workstations.

- It shall respect POSIX and ANSI standards whenever possible.
- It shall be stand-alone, i.e., self consistent. Users should not be required to download any other piece of software to use it. Additional software download may be required to extend functionality, but that should remain optional.
- As a data processing engine, eclipse is to run without user interaction or displays of any sort. These are left to external software packages. Stubs might be provided, e.g., to display an image or a plot, but they should never be mandatory in a command.
- Image processing or data analysis functionality is offered in a C library, accessible through a standard header (.h file) and a compiled library. Access is given to end-users through Unix commands.
- All Unix commands shall be documented with manual pages.
- A complete User's Manual and cookbook shall be available from the Web, and possibly with the software distribution.
- Only standard and widespread formats should be used for inputs and outputs.
- If eclipse replicates a functionality already present in another data reduction software facility, it must be justified either by speed gain, algorithm enhancement, user control over the algorithm, or ease of use.

3. Portability

As a portable library, eclipse should not only be able to be compiled on various platforms of today, but it also should be compilable on future platforms. Adhering to programming standards whatever they are (POSIX, ANSI, or simply coding conventions) ensures a reasonable survival of the library over time.

More generally speaking, writing portable software increases the quality of the code. By compiling the source with various compilers and running it on different platforms, weaknesses and defects are put under stress and are therefore revealed, whereas they might hide indefinitely or be very hard to find with a single development environment. It is a good idea to respect global programming conventions whenever possible.

Specializing a piece of software for a given platform is usually done for reasons of speed, to access the dedicated hardware or software present on the local system. Specializing software also means reducing its life expectancy, because underlying hardware or software may not be vendor-supported in the future.

4. File formats

The following file formats are supported by eclipse:

- FITS through an internal lightweight library
- PAF files: parameter files, internal to ESO
- ini files: windows-like parameter description
- PGM/PNM/PBM: to allow universal pixel translations
- various ASCII formats.

5. Coding standards

Coding style for eclipse conforms to that specified in the document "Recommended C Style and Coding Standards." Any eclipse developer should have most of the rules described there in mind when coding.

Following coding rules is often seen as a matter of taste. After all, if a programmer writes thousands of lines without indentation, it will also compile, and if it runs in the end, who cares about the style!

The philosophy behind coding rules and conventions is that a piece of software is defined by more than just the executable file that is produced after compilation. Pages of code are very often modified to correct bugs, enhance functionality, include into other (unforeseen) contexts, etc. It is vital that any developer in the team be able to modify somebody else's code with minimal overhead.

6. Modules

Most eclipse modules can be detached from the library to be reused as code snippets in other applications, thereby offering versatile functionality:

- File format handling: FITS images, cubes, tables, ini file handling, PNM output, various ASCII parsers
- Image processing primitives: Efficient bulk data loading, binary, integer, floating-point image handling, image filtering, both convolution and morphologic filters, 2-D cross-correlation, connected components (segmentation), cube and image arithmetic operations, dead pixel detection and correction, object detection, data extraction, flat-fielding with robust fit, image generation, statistics, photometry, image space resampling (warping), image combination, cube stacking with 3-D filtering
- Math primitives: Random number generation according to a given distribution law, FFT, curve fitting, matrices, fast median computation, point-pattern matching
- Unix handling: Portable endian detection and byte-swapping, message printing and logging, time measurement, compressed file reading/writing, generic dictionary handling, software configuration through environment variables, file locking, file system information query, command-line option handling, extended memory handling, memory-mapped file support through a generic interface, token-based ASCII file parsing, socket support, time printing in ISO8601 format, user identification
- Infrared specific: Database of 832 infrared standard stars included in the code, ISAAC-specific data reduction procedures.

7. Memory handling

Memory handling in eclipse has been carefully studied to avoid the usual limitations met in image processing software systems. The upper limit for allocatable memory is not fixed by the amount of RAM and swap found on the system, but by the amount of disk space accessible to the current user. The price paid

is degraded performance as soon as the system runs out of memory, but the gain is a uniform interface to the programmer who does not have to care at all about memory handling. The same source code will process a gigabyte of data no matter what the locally available memory.

8. Algorithms

Several ESO pipelines are now based on eclipse for number-crunching developments. This has produced a number of high-level algorithms specialized for data processing, for one instrument or another. Some of these algorithms are adaptable enough to be used for other instruments not initially foreseen (e.g., the high-level data reduction tasks written for CONICA, a VLT instrument, also perform very well on CFHT Adaptive Optics data).

9. Parallel processing

Since eclipse offers its functionality through a Unix command-line call, it is possible to parallelize it through coarse-grain parallelism. A master machine can send a list of Unix commands to run to a group of slave nodes and get the results back. Since eclipse is C-based, it is easy to add potential communication procedures (based e.g., on MPI or PVM) to make the algorithms fully parallel.

Wide-Field Imager (WFI) data from the 2.2-m telescope at La Silla is currently processed on the ESO Beowulf system using eclipse.

10. Python/eclipse

Like any other C library, eclipse can easily be interfaced to appear as a Python extension. This would allow image processing programs to be object-oriented, and to benefit from all the advantages of using Python.

The eclipse source code is currently under modification to make it easier to interface to Python. The wrapper code will be generated by SWIG and a Python-compatible procedure will be distributed to offer eclipse as a Python extension.