Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

The ROOT C++ Framework for Astronomy

Reiner Rohlfs
INTEGRAL Science Data Centre, Chemin d'Ecogia 16, CH-1290 Versoix, Switzerland, Email: Reiner.Rohlfs@obs.unige.ch

Abstract. ROOT[1] is a C++ framework developed at CERN, Geneva. It was designed for high energy physics to handle and analyze large amounts of data in a very efficient way. Its main features are a C++ - interpreter, graphical tools and an object I/O - system. The INTEGRAL Science Data Centre (ISDC)[2] is using ROOT as a graphical tool and as a scripting language in the off-line analysis. To use the full capability of ROOT for astronomical data analysis a link is required. Therefore we complete this system with a library called AstroROOT. With AstroROOT we can store astronomical data organized in tables and images in ROOT - files using the efficient I/O system of ROOT. Furthermore we are developing a new astronomical image display for the ROOT framework. Finally we can use the buildin C++ interpreter as a scripting language in an interactive session to analyze and display astronomical data.

1. ROOT

ROOT (Brun & Rademakers 1997) can be downloaded from the ROOT web page[3] as binary files for more than 20 platforms and compilers. These includes several Linux versions, HP - systems, IBM AIX, Sun SPARC, MacOS and WindowsXP/NT. And also the source code of the full system is available and can be compiled easily. ROOT supports several features and tools for high energy physics like an event generator, a detector simulator and an event reconstruction. But the most interesting sub - systems for astronomical data analysis are the built in C++ Interpreter, the graphical tools and a powerful I/O system which are described hereafter. In total ROOT is a C++ - library with more than 100 classes.

1.1. The C++ Interpreter

The ROOT C++ interpreter is a program by itself. It is also available in every compiled user program of ROOT. It is nearly 100 % ANSI C/C++ compatible: All standard C - functions are available. Due to differences between compilers

[1] `http://root.cern.ch/`

[2] `http://isdc.unige.ch`

[3] `http://root.cern.ch/root/Availability.html`

supported by ROOT it is not possible to define in the interpreter a class derived from a compiled class with virtual functions.

In an interpreter session it is possible to define variables of compiled ROOT classes and to call every compiled function of the C++ library of ROOT. With these features it is easy to read data from a file, manipulate them for example with standard C - functions, open a window with a ROOT class and display the data using one of the graphical tools described in the next chapter. Often these steps have to be executed again and again. To simplify this procedure the C and C++ statements can be written in a so called macro file which can be executed by the interpreter. When these macros grow and become a stable function they can be compiled with a C++ compiler because the language of the interpreter and the compiler is identical: C++.

Besides calling all ROOT classes and functions from the interpreter it is also possible to call user defined classes and functions in the interpreter. Within the ROOT package a program called rootcint is delivered which generates a dictionary code from the header files of the user source code. The interpreter uses this dictionary information to be able to call the user functions. The user code together with this dictionary code has to be compiled and linked into a shared library. The interpreter is able to load these shared libraries, find the dictionary information and immediately all exported user classes and functions of the shared library are available in the interpreter.

At ISDC we use the interpreter as a scripting language for offline scientific analysis. User functions to read and write FITS files (O'Neel et al. 2004) and a user class to call FTOOLS like programs are available. With these tools we can run within the interpreter astronomical scientific analysis of the INTEGRAL data.

1.2. The Graphical Tools

A set of C++ classes of the ROOT library are available to draw graphs, histograms and detector elements. Classes for basic elements like a line, an ellipse and text are also part of ROOT. Once a graph is drawn all attributes like color, size, text and makers can be modified interactively with a few mouse clicks. Three dimensional graphs can be rotated with a mouse movement to be able to visualize the data from different view points. Several curves with different attributes can be superimposed in one graph.

Furthermore fitting of graphs either with pre-defined or with user defined curves is implemented. Again this can be done either with a function call or interactively. A range on the X - axis can be defined before the fit is applied.

A graph with modified attributes exist only in memory. But with ROOT this modified graph can be saved as a C++ - source code in a macro file. With the help of the interpreter this macro can be read and the interactively modified graph is present again.

Besides these graphical tools, classes to build graphical user interfaces (GUI) are part of the ROOT - package. All widgets, like buttons, text and number entry, list box and window menus, are available. It is possible to combine the graphical tools and these widgets in one window. Unfortunately a GUI - builder is missing.

At ISDC we used these graphical and GUI classes to develop applications to visualize the scientific as well as the housekeeping data of the INTEGRAL satellite. Since the launch of INTEGRAL they are used in near-real time to monitor the satellite and the instruments of INTEGRAL (Rohlfs & O'Neel 2000). The user can select the data to be displayed and can modify the graphs interactively. Once a set of graphs in one or several windows are defined the configuration can be saved in a file using the I/O system of ROOT. In a next session the previously configured screen with several windows and displays can be reloaded by selecting just one ROOT file.

1.3. The I/O system

To store and to read efficiently huge amounts of data as they will be produced with the new experiments of high energy physics, ROOT introduced a new file structure and a new I/O system. The advantages of this system are:

- A high performance because only data required for the analysis are accessed.
- Efficient read functions with a great number of selection criteria.
- Individual buffers in a file are compressed by themselves and can be read without uncompressing the whole file.

The ROOT I/O system can store data of any C++ class in files. A program called rootcint automatically generates the relevant source code (streamer functions) to store objects in ROOT files. At a write operation this streamer function converts all data of a C++ class into a binary stream and recreates the C++ class during a read operation from this binary stream. For better performance one may store data from one object into several buffers that can be read independently. It is even possible to read data from a ROOT file when the source code is not available any more. Additional "streamer info" are stored in a ROOT file to be able to regenerate the source code of a saved C++ class.

2. AstroROOT

AstroROOT[4] is an extension of ROOT for astronomical data analysis. It consists of a set of C++ classes and of some executables as well as the ROOT interpreter with the capability to call all the additional classes and functions of AstroROOT.

First of all it is possible with AstroROOT to call all cfitsio functions within the extended interpreter. Now it is possible to read, write and update FITS files within the ROOT framework. With AstroROOT a set of container classes representing tables, columns and images with header information are available. These classes are more convenient than the cfitsio functions, because they have more powerful functions to access FITS files. As a result the user code is much shorter and simpler. For example, it is not necessary any more to allocate a buffer to read data from a FITS file. All this is done transparently inside the container classes. Another advantage is the possibility to store astronomical data not only in FITS files but also in ROOT files, using the I/O system of

[4]`http://isdc.unige.ch/index?Soft+astroroot`

ROOT. Storing data in ROOT files is more efficient than to store the data in FITS files, using tables with many columns or several extensions on one file.

With a program of AstroROOT, called param_gui, FTOOLS parameter files can be read and a GUI to edit the parameters is automatically created. With a few lines in the FTOOLS parameter file the GUI can be configured. Frames, tap-widgets and sub-windows can be defined. Comment lines with a special key are read by this param_gui program to build the GUI. With a button on the GUI the associated program can be started as soon as the user has defined the program parameters with the GUI. Furthermore an application function is available to open this GUI within a user executable.

An astronomical image display is in development to visualize astronomical images within the ROOT framework. The wcs - library[5] is used to display the coordinate systems. First successful tests with INTEGRAL images were performed.

3. Conclusion

Although ROOT was designed and developed for high energy physics it includes several features and C++ classes which also can be used directly for astronomical data analysis. The ISDC system takes the advantages of the graphical tools and the interpreter and has demonstrated that ROOT can be efficiently used for astronomical purpose. With AstroROOT we support the missing features to use ROOT as a complete system for astronomical data analysis. These are FITS file access in the ROOT interpreter and an astronomical image display within the ROOT framework. The second is still in development but already available as a beta version.

References

Brun, R., & Rademakers, F. 1997, Nucl. Inst. & Meth. in Phys. Res., A 389, 81-86

O'Neel, B., Peachey, J., Lerusse, L., Beck, M., Paltani, S., Walter, R. 2004, this volume, 193

Rohlfs, R., O'Neel, B. 2000, in ASP Conf. Ser., Vol. 216, ADASS IX, ed. N. Manset, C. Veillet, & D. Crabtree (San Francisco: ASP), 687 - 690, The ROOT ObjectOriented Framework to Analyze INTEGRAL Data

[5]`http://tdc-www.harvard.edu/software/wcstools/`