Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

Overview of the Ftools Software Development Philosophy

W. Pence

NASA/GSFC, Code 662, Greenbelt, MD 20771

Abstract. This talk will describe some of the features that have lead to the success of the ftools project and have enabled it to produce and manage a large package of software with low development costs. The ftools software package is a modular, platform-independent set of programs for analyzing FITS format data files in general, with a strong emphasis on data from high energy astrophysics missions. Ftools development began in 1991 and has produced the main set of data analysis software for the current *ASCA* and *XTE* space missions and for other archival sets of X-ray and gamma-ray data. One of the original requirements of ftools was to support both the IRAF and non-IRAF user communities, therefore the ftools software can be built either as an IRAF package or as a set of stand-alone executables that can be run directly from the host operating system. Platform-independence has been achieved in part by adopting FITS as the run-time data analysis format. Other external projects are now developing their own ftools analysis tools that can be layered on top of the existing ftools packages. We encourage further such collaborations as a cost effective way to develop new software.

1. Background

Development of the ftools software package started in 1991 at the HEASARC (High Energy Astrophysics Science Archive Research Center) to support analysis of data from the *ASCA* X-ray Satellite. During the past five years, ftools has grown into a large system containing both generic utilities for examining any data file in FITS format, as well as mission specific analysis tools for current and past X-ray and gamma-ray astrophysics missions. Because the ftools project could not afford to spend much effort on new systems-level development, it has adopted existing infrastructures when possible.

One of the original requirements for ftools was to be able to support both the PROS/IRAF and XANADU user communities, therefore we adopted the IRAF-style parameter file mechanism as the main interface between the application program and the user. This allows the software to be built as a package under IRAF, but in addition, the availability of a standalone parameter interface library, initially from SAO, also enables the ftools source code to be built as standalone executable programs. In this way we were able to satisfy both the IRAF and non-IRAF user communities.

The other major initial decision was the choice of analysis data format. We adopted FITS for several reasons, including the fact that most of our data files are already in FITS format, and FITS is a versatile format well suited for the type of "event list" data common in high energy astrophysics. FITS is also a good format choice because it is platform independent and files can simply be copied from machine to machine without any translation. This choice was only practical because the FITSIO (and later CFITSIO) interface library was available for application programs to easily read and write FITS files. Some initial concerns about the efficiency of using FITS formats (since FITS was originally developed as an archive and interchange data format, not an on-line analysis format) have largely been dispelled. Recent throughput tests of CFITSIO show that it can read or write FITS images or tables with rates in the range of 2–5 $\mathrm{MB\,s^{-1}}$ on current generation workstations which is very close to the limit imposed by the underlying magnetic disk.

The latest ftools v3.6 release now contains more than 220 tasks subdivided into a dozen packages (e.g., FITS utilities, images, timing, *ASCA*, *ROSAT*, *XTE*). The software is supported on a wide variety of operating systems (e.g., SunOS, Solaris, Alpha/OSF, Ultrix, SGI/IRIX, Linux, and VMS) and more than 200 registered sites worldwide have installed it.

2. Development Philosophy

The quality of the programming staff is of course one of the primary factors in the success of any software development project. All the ftools programmers are required to have a strong science background and most in fact have a M.S or Ph.D. in the physical sciences. Having a science background enables the programmers to understand the requirements of a new application program faster without the need for a minutely detailed set of requirements.

In a project like ftools, which lasts for many years, there will inevitably be some staff turnover, thus we have found it important to require cross training and overlapping duties to ensure continuity in case a key programmer leaves the project. We have also learned that it is getting more difficult to find and hire experienced FORTRAN programmers, presumably because fewer students in universities are learning FORTRAN. We have always supported both FORTRAN and C application programs in ftools, but we have shifted to more emphasis on writing software in C rather than FORTRAN as the skill mix of the programmers has evolved.

The ftools project has never had any formal requirements, design, or code reviews. Instead, we usually rely on the close interaction of an individual project scientist and a ftools programmer to develop the final product. In a few more complex or critical tasks, we may have a somewhat larger subgroup all working on a project, but the vast majority of ftools tasks have been single person projects. In effect, the ftools development is a distributed system with minimal central oversight or management bureaucracy. This approach does have some drawbacks in that the resulting ftools package is somewhat less homogeneous, and the quality of the code can be somewhat uneven. However, the increased productivity resulting from of this type of management more than compensates for this. The only formal management structure imposed on the project is a bi-weekly group meeting to maintain communication among programmers and scientists.

The ftools coding standards are mainly driven by the requirement to support the software on a wide variety of platforms and compilers. The primary coding requirement is to adhere to very strict ANSI C or FORTRAN-77. Even so, the differences between compilers continues to cause headaches and is a major expense in terms of having to build, test, and debug the software on all the different supported platforms. Because of this, we do not guarantee support for all possible compilers, but will at least support the GNU gcc and g77 compilers, which are freely available for most platforms. Because of these portability issues, we are currently reluctant to use FORTRAN 90 or C++ in any of the ftools software. If we do move to these languages in the future we will probably stick to GNU g++ (and g90 if it ever becomes available).

3. Code Management and Distribution Procedures

For most of the history of the ftools project there has been no formal configuration control of the software. Instead, we assigned one programmer to manage the "official" version of the software, and all the other programmers would deliver any changes to this person for installation. This worked satisfactorily, but in the past year we have switched to using the GNU version control software package called CVS. This turned out to have relatively low start up cost and now allows each developer to safely check-in new versions of their software whenever necessary. CVS also enables better tracking of different release versions of ftools and allows new enhancements to be checked in without affecting the frozen release version which may be undergoing final beta testing.

We rely on the Web and anonymous ftp to distribute the ftools software to users. From the ftools web page[1] users can download selected components of the ftools or the entire package. Currently we provide the source code and an intelligent build script that the user runs to compile and link the software. This works well in the majority of cases, but there are always a few users who experience difficulties, mainly due to configuration problems on the user's machine that are beyond our control. To better address these users, in the future we plan to distribute executable versions of the software, at least for the major platforms. This can introduce its own set of installation problems, but at least the user will not require a functioning C and FORTRAN compiler on their system in order to install ftools. In order to make the size of the executable files more manageable, a technique for combining several related ftools tasks into one executable program has been developed. This is transparent to the user but can reduce the disk space required for ftools by a factor of ten.

4. Support for External Ftools Packages

The success of the ftools project has lead other groups to consider adopting the same approach for their software development. Currently, several external X-ray groups have developed their own ftools-compatible packages, or are strongly considering doing so. We encourage further collaborative efforts like this, and view it as a very cost effective way for other projects to rapidly implement a new package of application software.

---

[1] http://legacy.gsfc.nasa.gov/ftools

The ftools group itself is continuing to make ftools more modular, so that it will become even easier for externally developed packages to be layered on top of the core ftools. The main requirements for developing a new ftools package are simply to learn to use the parameter file interface for communicating with the user, and the FITSIO or CFITSIO libraries for accessing FITS files. Most of the other necessary pieces can easily be implemented by copying the directory structures and the make procedure files that are in the current ftools package. The HEASARC is more than willing to work with any interested group in setting up their own ftools project.

5. The Future of ftools

The HEASARC is strongly committed to supporting and enhancing the ftools for the foreseeable future. We are currently starting development of a new analysis package for the "*Astro E*" X-ray mission (the successor to *ASCA*).

As well as developing the traditional C and FORTRAN based tasks which make up the core of ftools, we are also continuing to explore newer GUI applications, based on Tcl/Tk. Current examples of such applications are the Xselect and fv tasks in the ftools release. We are also considering ways to provide ftools-type analysis capabilities over the Web using Java.

The compatibility with IRAF is an important feature of ftools, so we are watching with interest the ongoing development of OpenIRAF. We fully expect to remain compatible with IRAF as it evolves, especially since the goals of OpenIRAF to provide better direct support for FITS files, and to decouple the IRAF tasks and software libraries from the CL, closely parallel the ftools philosophy.