*Astronomical Data Analysis Software and Systems IX*
*ASP Conference Series, Vol. 216, 2000*
*N. Manset, C. Veillet, and D. Crabtree, eds.*

# A C Programming Interface to the IRAF VOS Libraries

M. D. De La Peña, A. Farris

*Space Telescope Science Institute, Baltimore, MD 21218*

**Abstract.** We have developed a methodology for building C programming interfaces to the IRAF Virtual Operating System (VOS) libraries. The IRAF VOS is the fundamental programming interface to the IRAF system, providing the environment for the development of scientific applications. C interface functions for the more commonly used IRAF, STSDAS, and TABLES libraries, as well as the semi-automated interface generation mechanism comprise the C Virtual Operating System interface, CVOS.

## 1. The IRAF System

As a programming and analysis environment, IRAF provides a variety of capabilities. The system has a stable, well-tested, and efficient infrastructure, giving the programmer access to an extensive array of libraries and scientific applications. Since the IRAF system is layered upon the host system and encapsulates the host system dependencies, it is by design, very portable. The programmer has the ability to extend the functionality of the system by writing new applications in Subset PreProcessor Language (SPP), the native programming language of IRAF, or in Command Language (CL), the IRAF scripting language.

## 2. Why write applications in C?

The ability to write IRAF tasks in C gives the programmer access to an array of well-tested software tools for integrated development, and to many useful algorithms which are readily available in C. Large numbers of programmers are already experienced with C, mitigating the need to learn a specialized language (SPP) solely for the purpose of writing IRAF tasks. It also allows the IRAF system and applications to be extended with a widely-used, modern language, rather than the obsolescent SPP. However, with regard to the important issue of portability, a greater responsibility is incumbent upon the programmer to adhere to the ISO Standard for C.

## 3. The C Interface to the IRAF VOS (CVOS)

The CVOS is comprised of both pre-built interface functions which the programmer can use "out of the box" to begin writing C applications, as well as a semi-automated interface generation mechanism which can be used to create additional bindings to needed IRAF functionality. In general terms, the CVOS provides the ability to write C programs which can take advantage of community software, but which also can use the capabilities and functionality of the IRAF system. The C programs can be built as either host-level or native IRAF tasks. A host-level task is a plain C program which makes use of functionality in the IRAF libraries. A native IRAF task is a C program which is a fully integrated IRAF task and can be executed from the CL.

The CVOS interface transparently handles any datatype conversions and other inconsistencies between C and the IRAF environment. In addition, the CVOS performs the necessary initialization in order to use IRAF functionality. The CVOS supplements the error checking capabilities of the programmer by providing an error handler stack to be used for specialized error handling. The error handler stack can accommodate up to 32 customized error handling functions designed by the programmer. Despite the integration with IRAF functionality, the CVOS retains the use of C standard language libraries, host-system libraries, and system utilities for software development.

## 4. Pre-built Interfaces

Pre-built C interfaces exist in the CVOS for the majority of important IRAF, STSDAS, and TABLES libraries and tasks. The libraries currently available through pre-built CVOS interfaces are listed in the Table 1.

Table 1. Pre-built Interfaces.

| Library Name | Package Origin | Description |
|---|---|---|
| clio | NOAO | IRAF Command Language interaction |
| curfit | NOAO | linear least squares curve fitting |
| gflib | STScI | tools for the manipulation of GEIS files |
| gsurfit | NOAO | linear least squares surface fitting |
| iminterp | NOAO | image interpolation |
| imio | NOAO | image access |
| mwcs | NOAO | mini-world coordinate system |
| nlfit | NOAO | non-linear least squares fitting |
| qpoe | NOAO | interface to position order event files |
| selector | STScI | syntax for access to data stored in FITS tables |
| surfit | NOAO | surface fitting |
| synphot | STScI | synthetic photometry |
| tables | STScI | tools for TABLES/FITS/ascii tables |
| vops | NOAO | vector (array) operations |
| xtools | NOAO | miscellaneous tools |

In order to use the CVOS to create C tasks, the application source code must include a CVOS system header file which contains critical declarations and definitions needed by the C interface functions and source programs. In addition, the C application source code must also include any library header files which correspond to the libraries/packages listed in Table 1; the library header files contain the prototypes for functions used in the application code.

## 5. Generation of C Interfaces

The CVOS provides a semi-automated C interface generation mechanism which can be used to create a C binding for any public SPP routine. A user-defined interface definition file, based upon the SPP routines for which the C bindings are needed, provides the necessary information to drive the CVOS generation program. Typically, this file consists of a one-line entry for each SPP function; the entry includes the return type, function name, and formal function parameters. Some parameters many need to be further annotated for input/output mode, possible SPP pointer conversion, and string length specification. The details of the interface definition file are too lengthy to be discussed here, but full information can be obtained from `http://ra.stsci.edu/CVOS.html`. All necessary datatype conversions between C and SPP are performed automatically by the generation program. The generated C bindings conform to ISO C standards and support both pass-by-value and pass-by-reference. Both the C binding header ("*.h") and source ("*.c") files are created by the interface generation tool.

## 6. Host-level Tasks versus Native Tasks

Special code must be incorporated into the C application source, and certain compilation switches need to be set, depending upon the way the executable is to be built – either as a host-level or as a native IRAF task. Host-level C tasks are programs similar to any pure C program, but can use the functionality in the IRAF libraries. In this sense, the IRAF libraries are no different than any other public library accessible to C. Host-level tasks are designed to be executed at the host-level with command line arguments, and therefore, they cannot be run directly from the IRAF CL. Since host-level tasks are effectively independent from the IRAF environment, they do not have access to IRAF environment variables. Any needed environment variables (e.g., shortcuts for directory pathnames) must be set at the host-level by the user.

In contrast, a C program compiled as a native IRAF task can take advantage of all the capabilities of the IRAF system. Not only do native IRAF tasks have access to the IRAF library functionality, but they can be run directly from the CL. Input/output and other information used to customize the functionality of a native task are handled by task parameters as is done with IRAF SPP tasks. Table 2 contrasts the attributes of a C task compiled as a host-level and as a native IRAF task.

## 7. Issues

While having the option of writing C tasks for the IRAF system provides a great deal of flexibility for the programmer, the issue of portability becomes a major concern. Although there is an ISO Standard C, the programmer must take the responsibility to adhere to this standard when writing C code. External C

Table 2. Contrast of Host-Level and Native IRAF Tasks.

| Issues | Host-Level | Native IRAF |
|---|---|---|
| C library | standard system LIBC | IRAF LIBC |
| Able to use IRAF functionality? | yes | yes |
| Execute from... | host command line | IRAF CL |
| Input values via... | command line arguments | parameters |
| IRAF parameter handling capabilities? | no | yes |
| Set environment variables... | at the host level | in IRAF |
| Special wrapper routine needed? | no | yes |
| Handle calls to exit() | yes | no |

libraries can be used to supplement functionality, but this could also potentially pose a portability problem.

Depending upon how the C task has been compiled, the version of the C library and the exception handling mechanisms invoked are different. Native IRAF tasks use the IRAF C library while host-level tasks use the Standard C library. Also, exception handling for native IRAF tasks is handled by the IRAF default exception handler. If the IRAF default behavior is not deemed sufficient, IRAF provides the functionality to post a customized exception handler. However, host-level tasks which encounter exceptions perform basic clean-up and exit to the host system.

## 8. Status

The CVOS is a library which is distributed as part of the STSDAS environment, and serves as the basis for the HSTIO library used to support the HST calibration pipelines for STIS, NICMOS, and ACS. The CVOS and HSTIO are also used by all new analysis tasks developed by SSG which are written in C. Information regarding the CVOS can be obtained from `http://ra.stsci.edu/CVOS.html`.