AIPS Memo 83

Dual Libraries and Binaries in AIPS

Patrick P. Murphy

July 19, 1993

Abstract

One of the most frustrating aspects of working with *AIPS* on a daily basis is dealing with the debug and/or optimization level built into the many libraries. There is inherently a conflict between the desire for high performance and the need to access symbolic debugging information, at least on many compilers. This is true on current versions of SunOS where one cannot use debug and optimize qualifiers together on compile or link commands. To address this problem, a scheme similar to one first proposed by Mark Calabretta ([1], [2]) has been implemented for the Sun-4 architecture in the `15JUL93` (formerly `15APR93`) test version of *AIPS* at both Charlottesville and AOC sites within NRAO. The rest of this memo describes in some detail the mechanisms used in this system.

1 Introduction

The basis of the system is to have two separate areas for object and/or shareable libraries: one that will hold the debug-capable, non-optimized versions of the object modules, the other for the optimized version with no symbolic debug information included. While the `$LOAD` area is still the repository for binaries (executables) compiled with the non-debug libraries, the intent is for the debug libraries to be used when creating local versions of tasks for debugging purposes, usually in one's home area or a subdirectory thereof. Several of the Unix shell scripts have to be modified to accomodate this change. The entire system is enabled or disabled based on the presence or absence of a single file: `$SYSLOCAL/DOTWOBIN`.

The new area is pointed to by an environment variable ("logical") `LIBRDBG`. These are defined in the generic portion of the `AIPSASSN` files as a directory parallel to the `LIBR` area, *i.e.*,, `$AIPS_VERSION/$ARCH/LIBRDBG`. If the `COMRPL` script described below fails to find this area and the `$SYSLOCAL/DOTWOBIN` file is present, the directory will be created if possible.

In the sections that follow, the behaviour described is for the most part that relating to files that reside *in the main AIPS directory tree*. Local files, *e.g.*,, those residing in a user's private directory, will still compile and link in the same way they always have. Throughout this document, the term "local" will be used to refer to such private files, and "system" to refer to files in the `$AIPS_VERSION` directory tree. The only significant changes users developing such "local" code really need to be aware of is the presence of a debug set of libraries, and the new command line option on `LIBS` (see below).

2 COMRPL – New Behaviour

The normal behaviour of `COMRPL` is to (a) compile a source code module into an object file, and (b) optionally move it to a holding area for eventual insertion into a library. Part (b) is not done for "local" code. The revised `COMRPL` procedure will compile any "system" module twice as follows.

First, any optimize/debug directives from the command line are removed, and any `PURGE` directive is overridden (the preprocessed source code will be needed for a second compilation). Thus, for these "system" files, the debug/optimize settings are determined solely by the `FCLEVEL.SH` and `OPTIMIZE.LIS` files for Fortran modules, and by `CCOPTS.SH` for C language modules.

The first compilation done will produce an optimized, non-debug object file. There is an inherent assumption here that the settings in the various files mentioned in the previous paragraph will produce just this.

This second compilation takes the basic options (again, exclusive of command line directives) and overrides them with `DEBUG`, `NOOPT`, and `NOPURGE`. Then both this object file and the first one are moved to their respective directories under the `LIBR` and `LIBRDBG` areas. The procedures will give verbose details on how the object files are kept separate, but knowledge of these details is not necessary for an understanding of the logical behaviour of the script.

3 LIBS – New Command-line Option

The use of the `LIBS` shell script is solely to generate options files for "local" compilations and links. In the revised setup described here, it is necessary to be able to specify whether the debug or non-debug versions of the libraries are used. This is accomplished now via, *e.g.,*: `LIBS $AIPPGM DEBUG` or maybe `LIBS $APGNOT -d`. Either form of the option is case insensitive. If any form of this is detected, the procedure will generate a list with the `LIBRDBG` set of libraries. Otherwise the default action will be to use the `LIBR` libraries.

4 COMLNK – New Behaviour

This revised procedure is similar in operation to `COMRPL`. Its behaviour depends on whether or not the `DEBUG` option is specified in the command line. If it is not, the behaviour of `COMLNK` is unchanged.

However, if `DEBUG` is specified, the program in question will be compiled and linked against the debug libraries. While the binary will be placed in the `$LOAD` area for "system" compilations/links regardless of the `DEBUG` setting, the intention is for the debug mode to be used for "local" use. The original version of the scripts (and of this document) used a separate `$LOADDBG` area for "system" debug binaries, but this use was abandoned prior to the public release of the scripts.

5 Impact on Shared Libraries

The use of shared libraries in the Sun (and now HP) versions of *AIPS* has been controlled for some time by the presence or absence of the file `$SYSLOCAL/USESHARED`. This has not changed with the modifications described above, except that shared libraries will only be used for the optimized, non-debug case. They are then generated in addition to the regular static libraries.

The location and naming of these shared libraries has, however, changed. The static libraries will remain in the various subdirectories in the `LIBR` area, but now the shared libraries will be stored directly in `LIBR` and are named after the area in question. Thus, what was formerly `$LIBR/APLSUB/SUBLIB.so` will now be found in `$LIBR/APLSUB.so`. The rationale for this change is to permit the easy movement of the location of the `AIPS_ROOT` area without necessitating a rebuild of a shareable library *AIPS* installation. It may also facilitate a binary distribution of *AIPS*. Both of these are possible by adding the `LIBR` directory to the SunOS environment variable `LD_LIBRARY_PATH`.

References

[1] "Disk usage, build time, and execution time for *AIPS* in SunOS under a variety of compilation modes",

Mark Calabretta, Australia Telescope National Facility, 1991/Oct/04, on the *AIPS++* Tools and *AIPS* Bananas E-mail exploders.

[2] "Verification", Mark Calabretta, Australia Telescope National Facility, 1991/Oct/04, on the *AIPS++* Tools E-mail exploder. See also the follow-up messages.