IRAF NEWSLETTER

September 1987 Number 2

Central Computer Services National Optical Astronomy Observatories* P. O. Box 26732 Tucson, AZ 85726

Table of Contents

* Operated by the Association of Universities for Research in Astronomy, Inc. (AURA) under contract with the National Science Foundation

IRAF STATUS AND PLANS - FALL 1987

1. Current System Status

IRAF has been in use within the general astronomical community for one and a half years now, since the first limited public release in February 1986. There have been three major releases of the system during this period, plus a number of limited releases to support new ports, or to get new software out to user sites that urgently needed it before the regular release became available. The major releases were as follows.

V2.3 The first limited public release, February 1986.

V2.4 Summer 1986. Provided support for SDAS and the STScI group image format, plus the first release of the SGI device independent plotter interface.

V2.5 Summer and fall of 1987 (current release). V2.5 provides support for a number of new host systems, including Sun workstations, the Microvax under either Ultrix or VMS, DG AOS/VS, the Alliant vector minisuper, and several other workstations, with a large number of new ports either in progress or planned (e.g., Convex, Apollo, HP). Device interfaces for the most widely used graphics terminals and plotters are now available, with new ones being routinely added as needed. An IRAF Hotline and site support group have been set up to routinely support installation and use of the system by the community. While V2.5 includes quite a bit of new software, most of the changes made in V2.5 have been minor bug fixes, minor tuning and efficiency enhancements, or changes made to support all the new ports, e.g., for vector graphics on workstations. There were literally hundreds of significant revisions in V2.5 in the system software alone, many occurring as a result of feedback from user sites (gripes and suggestions as well as bug reports).

The V2.5 release is currently in progress, and we strongly recommend that all sites which are actively using IRAF request and install the new version. Sites calling in problems on the IRAF Hotline which have not yet upgraded to V2.5 will be encouraged to upgrade, since it is likely that any problems experienced with an old release will have been fixed in V2.5. A summary of the major revisions in V2.5 is provided elsewhere in this newsletter, and detailed revisions notes are available upon request and are (or will be) automatically provided to anyone receiving the V2.5 release.

To date IRAF has been shipped to something like 150 sites, many of which are actively using the system, with the V2.5 release still underway. Nonetheless we continue to maintain that IRAF is not yet ready for general release, chiefly because the general image processing facilities are still under development (NOAO data reduction comes first), the programming environment is not yet sufficiently well documented, and we have not yet developed our release procedures and distribution and support system to the point where we can efficiently support the number of systems anticipated for the general release. Sites considering getting IRAF should first contact us (J. Barnes) to discuss their needs and the availability of and capabilities provided by the current system.

2. Current System Size

There are many ways of measuring system size, e.g., the approximate size of the system in terms of lines of code, and the runtime size of the system in terms of the amount of disk and memory used. Measuring system size in lines of code can be difficult because it is difficult to define what a line of code is, or what it has to do with system size. I recently ran across a technique for measuring system size in lines of code that I like because it is simple, and measures the true information content of the system in normalized units (the technique was mentioned in an AIPS project paper).

The idea is to measure the total size of a text only copy of the system, i.e., excluding all objects, executables, and other machine generated binary files, but including all text files, e.g., source files, help files, parameter files, and so on. We then assume an average number of bytes per line of text to get the system size in lines of text. Using WTAR to make a text only archive of IRAF, including the new CCDRED and APPHOT packages (not included in the V2.5 release), and estimating about 10K lines each for the system dependent parts of the AOS/VS and VMS HSI's (host system interfaces), I get the following numbers for the V2.5 IRAF system size. It is important to note that we have made an effort

to keep the system size down, by extensive use of modularization and layering (common libraries etc.), rather than duplicating and modifying code to build new programs. All but a few thousand lines of this code is completely portable and device independent.

text-only system size:	27 Mb
lines of text:	820K lines

This information is distributed over something like 7000 files in over 250 directories. Disk usage depends upon the host system and upon whether or not the system has been "stripped" of all non-runtime sources and other files. A stripped system ranges in size from about 18 Mb to 25 Mb. A full system can be as large as 60 Mb. Further space savings are expected in the future as shared libraries become more widely available in UNIX systems. Runtime memory requirements depend heavily upon the application but are generally modest for an image processing system, e.g., 1-2 Mb per user.

These size estimates do not include any software produced for the IRAF environment by sites other than NOAO, e.g., at STScI, CFA, DAO, and so on. We anticipate that eventually much or most of the software available for use within IRAF will be produced by sites other than NOAO.

3. Host Systems Currently Supported

The number of host system configurations for which IRAF is available is growing rapidly. Despite the number of host systems supporting IRAF, there is always only *one* version of the main IRAF system running on all those systems (except for a few files in the host system interface on each system). Since most of IRAF compiles and runs without modification on any IRAF host, it is relatively easy to keep each system up to date. This is of critical importance to be able to support IRAF on such a range of systems, with each host environment and IRAF itself continually evolving.

The current IRAF implementations, presented roughly in the order in which they were implemented, is as follows, including notes on the current extent of the support available for each system.

BSD Unix Fully supported at NOAO under 4.3BSD Berkeley Unix. The master IRAF software development machine is a BSD system, although most use of IRAF now takes place on other systems.

VMS The original port of IRAF to VMS was carried out by STScI during 1984 and 1985. VMS/IRAF has been maintained by NOAO since the fall of 1985. The VMS version of IRAF is fully supported under the latest version of VMS (currently 4.5).

Sun Unix Fully supported at NOAO under the latest version of SunOS (currently 3.4). SunOS changes a lot in every release (as new hardware and software is introduced), and continual support is necessary to track these changes. IRAF is available for the Sun-2 series, is fully supported on all Sun-3 machines, and will soon be supported on the Sun-4 series (the RISC machines). Graphics and image display facilities, including a primitive image cursor readback capability, are currently available under SunView, and support for X/NeWS and/or X itself is planned.

MicroVax The microvax is fully supported by NOAO under both Ultrix and VMS, using hardware and software provided courtesy of DEC for this purpose. We have a GPX microvax with two system disks, one running Ultrix and the other VMS. We alternately boot up either system to update or test the microvax versions of IRAF, or to generate distribution tapes. Our Ultrix is currently version 1.2, but we will soon upgrade to 2.0. Support for the GPX is limited at present, as we are still waiting for Version 11 X windows to be fully supported on these systems. A limited vector graphics capability (Xterm) is nonetheless currently available under Ultrix, and a prototype image display capability (Ximage from CFA) should be available soon.

AOS/VS The AOS/VS version of IRAF was produced and is maintained by Skip Schaller at Steward Observatory, and is supported under a collaborative arrangement with NOAO and Steward (Steward is right across the street from NOAO).

Alliant IRAF was ported to the Alliant in 1986 by NOAO and is currently maintained by Dennis Crabtree at the Dominion Astrophysical Observatory (DAO) in Canada. Alliant/IRAF is supported under a collaborative arrangement between DAO and NOAO.

ISI IRAF was recently ported to the ISI (using the Greenhills C and Fortran compilers) by Richard Stover at Lick Observatory. The degree of support which can be expected for this system is unknown.

Masscomp IRAF was recently ported to the Masscomp system by Rob Seaman at the Univ. of Wyoming. The degree of support which can be expected for this system is unknown.

Support for systems like Data General AOS/VS and the Alliant, where NOAO does not have direct on-site access to a machine to support IRAF, is under a collaborative arrangement with the remote site and NOAO. We work closely with these sites to help keep IRAF up to date and as well tested as possible on the target machine. Requests for an IRAF distribution for systems supported in this way should be directed to NOAO in the usual fashion, i.e., by first contacting us to obtain the order form and then filling out and returning the form for entry in our site database. Distribution kits are normally mailed from NOAO, to simplify bookkeeping and since the documentation is maintained here. Help with installations and device interfaces is available from the IRAF Hotline and may also be available from the remote technical support site. Requests for help using the IRAF applications software should be directed to NOAO.

3.1. New Ports

In addition to the above, a number of new ports are in progress, are planned, or are actively being discussed. Systems for which ports are planned or in progress include Apollo, Convex, and HP. A generic UNIX System V version of IRAF is planned, and will be used as the basis for the HP port, and similar ports to other System V based variants of UNIX. There has been a lot of talk about a port to the Macintosh II, but we are still waiting for Apple to put together a system based on the Mac II which is capable of supporting IRAF (e.g., the UNIX is not quite ready yet). Whether or not such a system is ever developed for the Apple, an IRAF-PC class machine for less than $10K is bound to become a reality within the next year or two. Powerful diskless nodes capable of high resolution graphics and image display are already available in this price range.

Many other ports have been discussed, including to some very interesting high end vector machines, but most lack sufficient commitment at this point to be worth mentioning here. Mention here is no guarantee that a port will actually be performed in any case, or that support will continue to be available once the initial port is performed. We recommend that sites considering running IRAF on a machine not currently supported contact us to discuss how difficult it is likely to be to carry out the port. The systems integration effort required for a good port should not be underestimated, but on the other hand excellent new systems are appearing on the market all the time, and we do not think that the effort required to port IRAF and subsequently support it on a new machine should prejudice users toward the choice of any particular vendor.

3.2. Device Interfaces

IRAF currently provides good device independent interfaces for all classes of devices except image displays, and an effort is underway to define such an interface (the IDI/IDK standards project). Devices like video terminals, graphics terminals, plotters, printers, magtape devices, file storage devices, network interfaces, and so on are already easy to interface to IRAF. Since our goal is to make new devices as easy as possible to interface (without sacrificing efficiency or functionality), and since so many terminals and plotters, etc., are already interfaced, there is little point in listing them here. Performance, price, and features are the most important considerations when purchasing new devices to be used with IRAF. Contact us if you are not sure what features to look for.

4. Systems and Applications Software Development

Since the first release of IRAF a year and a half ago, maintenance and support of the system has grown to the point where it now consumes a major fraction of the available manpower, competing with important new software development projects for limited resources. Our solution to this threat to continued system development has been a big push to get the existing system in as good a shape as possible, so that a lot of sites can install it and use it with minimal problems and support needs, so that we can go off and work on *new* systems software for a year or so.

The technical site support group (IRAF Hotline etc.) is now functioning routinely, and V2.5 is frozen and in full scale distribution to the user community. New ports will continue to consume limited system development resources, as will system management and planning, but the main thrust for the next year or so is going to be another major phase of systems development, concentrating on the topics of interactive image processing, display devices, and workstations. Some work has been done in these areas in V2.5, but much important work remains to be done. These projects have been delayed long enough and we feel that it is critically important that they go forward now, even if competing projects and requests have to be delayed.

We are also entering a new period of IRAF applications software development. Up until now we have mainly been playing catch up, coding data reduction facilities in IRAF to duplicate (or improve upon) similar capabilities in older systems which we could no longer support on the new computer systems for one reason or another. With most of the critical items out of the way, it becomes less obvious what to implement next, and it is time to get the greater IRAF user community involved in project planning. As a first step we will be sending out a questionnaire to all people on the IRAF newsletter distribution list. For the longer term there are perhaps more significant things we can do to help get the user community more involved in the project. Some ideas are given at the end of this article.

5. Software Development by Outside Sites

A major development with IRAF in the past year has been the rapid increase in IRAF based software development by sites other than NOAO. We are very pleased to see this happening, since if the trend continues it means that the scientist using IRAF will eventually have a great deal of data analysis software available within a single standard data analysis environment, regardless of where they go or what host machine they wish to use. The advantages to software developers using IRAF are that the most commonly needed system facilities are probably already provided by the environment hence do not need to be developed, software portability and device independence are automatically provided by the environment, easily solving what could otherwise be a major problem, and the proven efficiency of IRAF VOS (virtual operating system) provides some assurance that their applications will also be efficient.

The first development in this area was the decision by STScI late last year to fully integrate several of their large software systems, including the SDAS software, into the IRAF software environment. Since that time the ROSAT group at CFA has begun developing the data analysis software for ROSAT within IRAF, and Dennis Crabtree at DAO (the Dominion Astrophysical Observatory) has announced that he will be developing an IRAF version of the highly successful DAOPHOT digital stellar photometry package, working with Peter Stetson of DAO, the author of the software. Several other projects have since announced that they will be providing analysis software within IRAF, and we have had many consultations with other groups considering doing so.

6. Increasing User Involvement in IRAF

We would like to see the involvement of users in the IRAF project increase in coming years. To help get the user community more involved, we could start by increasing the communication between the IRAF project and the user community, as well as between IRAF users, e.g., by setting up an electronic bulletin board or some sort of network news facility, and by hosting public workshops and tutorials on using IRAF and on programming in IRAF. A database of user contributed software could be set up for access via a network or dialin, as has been done very successfully recently for math library software, or we could at least provide a bulletin board so that such software could be announced, or requested (e.g., has anyone written software to do X already?). As use of the IRAF

programming environments for user contributed software grows, such facilities will become increasingly desirable.

Publishing the IRAF newsletter on a fixed schedule is our first step in this area. Implementation of some of the remaining projects may have to be deferred until the critical next round of systems development projects are completed, to avoid further delays in these projects.

Doug Tody

NEW NEWSLETTER POLICY

Beginning with this issue of the IRAF Newsletter, we will no longer be including detailed software revisions notes in the newsletter. Publication of the newsletter will move to a fixed schedule, rather than trying to couple the newsletter to major IRAF releases. This change was necessary because the revisions notes for IRAF are too voluminous to be included directly; we will include a summary of the major revisions instead, sending the detailed revisions notes out with the actual software distributions, along with a list of known bugs in that release. Coupling the newsletter to major releases is increasingly difficult, as we may have different release schedules for different systems. Most importantly, we would like to publish the newsletter more frequently than we have in the past, and the easiest way to do that is to publish it at fixed intervals of 2, 3, or 4 times a year. The exact interval at which the newsletter will be published has not yet been decided.

Doug Tody
Jeannette Barnes

PUBLIC RELEASE OF IRAF VERSION 2.5

IRAF version 2.5 is now ready for public distribution. All sites currently running IRAF are encouraged to upgrade to this new version. This new version of IRAF can be obtained simply by completing and mailing to us the attached IRAF order form. The return address is on the order form. We also ask that current sites include with the order form the magnetic tape(s) that were previously sent to them with their current IRAF distribution. Sites that have already received IRAF version 2.5 but have not yet returned their old system tapes are invited to do so at any time in anticipation of the next release.

Please be sure to fill out the order form with ALL the current site information that is requested. Information about operating system version numbers, hardware components, etc. help us determine what system we should send you for the easiest installation, as well as help us detect possible noncompatibilities. Sites requesting more than one IRAF system distribution are asked to include a form for each system.

The "load and go" IRAF distribution is simply a snapshot of the complete IRAF system. The "you relink" distribution is the same minus the executables. Sites that should request "load and go" systems include VMS sites running 4.4 or later, Unix sites running Berkeley Unix 4.3, Ultrix 1.2 and 2.0 sites, Sun 3 sites running 3.2 or later, and all AOSVS sites. If you are in doubt about the distribution system that you should request, please indicate your uncertainty on the order form and we will attempt to send you a system based on the site information supplied on the form.

There will be no charge for this distribution or for the accompanying documentation. However, charges are being considered for forthcoming releases. We will keep you informed.

Jeannette Barnes

SUMMARY OF REVISIONS IN IRAF VERSION 2.5

The following is a brief summary of some of the new features available in IRAF version 2.5. A complete list of all system and package revisions is being prepared and will be included in all future distributions, as well as mailed to all sites that have already received a V2.5 IRAF release.

This release concentrated primarily on optimizations, bug fixes, and tuning of the V2.3 system, rather than on the addition of major new features or facilities. Considerable work has gone into the SUN/IRAF system, which is now in use at the CTIO and KPNO telescopes for data reduction while observing, as well as within the community for normal IRAF use. The Microvax is now supported under both Ultrix and VMS. The hardware and software upgrades, including a GPX upgrade, necessary to support IRAF on our Microvax, were donated by DEC expressly for the purpose of supporting IRAF.

First, and foremost, all users using V2.5 for the first time must do a `mkiraf`. The syntax for the old `login.cl` file is no longer current. You are advised to delete all of the existing parameter files, as well, unless you feel strongly about not doing so. If you do not delete these files, then be forewarned that some tasks may require an 'unlearn' to have them execute properly.

- IMFORT, a Fortran programming environment intended for use primarily by scientists wanting to interface their own host Fortran programs to IRAF, is available with V2.5. The IMFORT interface provides complete access to IRAF images, including header information (only IRAF format images are currently supported), basic command line argument i/o, and miscellaneous other facilities suitable for the construction of simple image processing programs. See *A User's Guide to Fortran Programming in IRAF: The IMFORT Interface*, by Doug Tody, in Volume 1A of the *IRAF User Handbook.*
- Full graphics support is available on the Sun workstation as well as limited support for image display including a rudimentary image cursor readback capability.
- Shared libraries have been installed on the VMS/IRAF system, making the system somewhat more efficient, as well as reducing the amount of disk space consumed by the system. The implementation of shared libraries used in V2.5 was done by Dennis Crabtree of DAO while on leave at STScI.
- Command mode is now the default everywhere in CL command streams. Simple scripts can now be written using the same syntax that you use when typing a command line at the terminal.
- Many of the tasks now pipe their cursor help pages to the task `page` - no more cursor options scrolling off the screen. Some of the `onedspec` tasks have not yet been converted.
- Your `login.cl` file now executes a file called `loginuser.cl`, if it is found in the login directory at login time. This file can be used to customize your login, the advantage being that the file is not modified when you do a `mkiraf.` Examples of its use are given elsewhere in the newsletter.
- The task `stty` must be used to set the terminal type. For example, if executing `stty` indicates that your terminal type is a VT640 but your terminal is, for this session, a VT240, then you type `stty vt240` to change the terminal type. Your `loginuser.cl` file can also be edited to reset the default terminal type. It is no longer necessary to explicitly set the `stdgraph` environment variable.
- The `eparam` parameter set editor task now supports a number of "colon" commands, e.g., exiting with a `:go` causes the task to be executed with the current query and hidden parameters.
- The 'q' keystroke is now the universal exit from all graphics cursor loops - including `implot`.

- A new keystroke '=' is synonymous with ':.snap' when you are in cursor mode. The output goes to the default `stdplot` device, or the last device specified in a `:.snap` *device* command.
- The size of the user area in memory that holds the image header information may now be specified via an environment variable - tasks that write header information at their time of execution may give a warning message that this area is full. The size of the user area may need to be increased for the tasks to execute properly. See the accompanying article *FITS UPDATE* for more information on this topic.
- WFITS has a new parameter for the blocking factor - it can now write FITS tapes that have 1 to 10 times 2880 bytes per block.
- The IMAGES package has undergone some modifications. The task `geodistran` has been modified and renamed `register`. The geometry tasks, `rotate, imlintran, register,` and `geotran`, have been made more efficient. A new task called `blkrep` has been added that will block replicate images. `imarith` has been modified to provide replacement of divisions by zero with a constant parameter value. The task `minmax` now prints out the minimum and maximum pixels as well as their values.

NOAO package revisions include the following.

A new package has been added to `noao.imred` called DTOI. This package contains a set of tasks for determining and applying a density to intensity transformation to photographic data. See the accompanying article in this newsletter by Suzanne Jacoby.

A new task has been added to `noao.astutil` called `pdm`. It is used to find periods in light curve data via phase dispersion minimization. See the accompanying article by Dyer Lytle for more details.

A new task called `powercor` has been added to the `noao.onedspec` package. This new task allows the user to correct KPNO/IIDS data for any effects due to the non-linearity of the instrument. For details of this correction, see NOAO Newsletter #6, June 1986, *How Linear is the IIDS*, by Phil Massey and Jim DeVeny.

Several other additions/changes have been made to the `onedspec` package. The parameters for the starting wavelength and the wavelength interval in the task `dispcor` can now be input from a file, if the user chooses - this change should help expedite echelle reductions. It also affects other packages in the `imred` package that call this task. A new keystroke 'v' has been added to `splot` for measuring line centers and equivalent widths. Improvements have also been made to the functionality of the keystroke 'h', and changes have been made to the output line for 'd' and 'e'. A major reorganization of the line lists and calibration files used by many of the `onedspec` and associated packages has taken place. Please see the accompanying article for more details. A new task called `shedit` has been added to this package as a first attempt to aid in the editing of header parameters for processing data within the `onedspec` package.

The `response` task in `noao.twodspec.longslit` has been modified to allow for specifying the image section to be used for determining the response as well as the image section to be used for deriving the normalization spectrum.

A new task called `ndprep` has been added to the `noao.proto` package. This generates neutral density filter calibration images using filter curves stored in the `onedstds$ctio` directory. The `imfunc` task in this same package now has three options: log base10, antilog base10, and square root.

The `noao.twodspec.apextract` package has undergone a major overhaul. The user interface has changed along with the task names, although the extraction algorithms remain the same. The package now depends heavily on the new shared parameter set concept introduced with version 2.5. A new task called `apnormalize` has also been added to the package. The purpose of this task is to remove the general shape from the flat field aperture spectra before dividing the data by the flat field. The flat field aperture spectra are normalized by a one dimensional normalization function derived by

extraction of a spectrum from an associated normalization aperture. See the on-line help pages for more information as well as the paper by Frank Valdes, *The IRAF APEXTRACT Package.*

A new package called `specphot` has been added to `noao.imred`. This package loads tasks and sets parameters in some of these tasks for reducing 2-dimensional spectrophotometric stellar data to flux calibrated 1-dimensional data.

Doug Tody
Jeannette Barnes

SITE SUPPORT AND THE IRAF HOTLINE

Beginning with the release of IRAF V2.5, we are establishing an IRAF Hotline to be used for questions relating to installation problems, device configuration, apparent bugs, and the like. These matters were handled by Steve Rooke in prior IRAF releases. Steve will continue to be involved with site support, but will now share those duties with another IRAF group member, Suzanne Jacoby.

As in the past, if your question is about how to use an applications program, or how to use IRAF to reduce your data, you should contact Jeannette Barnes. The Hotline should generally be used for system problems, not user or applications program problems (when in doubt, just guess or call the Hotline anyway).

An answering machine has been installed for round-the-clock coverage. We will try to return calls or otherwise respond within 24 hours during weekdays, though often a problem may be resolved within an hour or two. Please be ready with your name and telephone number, the name of your institution, the version of IRAF you are running, your operating system and version, computer hardware, and a brief description of the nature of the problem.

IRAF Site Support		
System & Installation Problems	IRAF Hotline	(602) 323-4160
Hardware, Ports	Steve Rooke	(602) 325-9399
User & Science Applications Questions	Jeannette Barnes	(602) 325-9381
Requests for Distribution, Documentation	Jeannette Barnes	(602) 325-9381
Bug Reports, Comments, Suggestions	Electronic Mail	*

*See accompanying article on email addresses for IRAF.

Steve Rooke

DENSITY TO INTENSITY CALIBRATION PACKAGE

A new package called DTOI (Density TO Intensity calibration) has been added to Version 2.5 of IRAF. This package contains six tasks for computing and applying a density to intensity transformation to photographic data. These tasks interactively compose an HD curve from measured densities of calibration spots and then transform a density image accordingly. Communication between the tasks is via a text file which the user can inspect or modify.

The HD curve, or characteristic curve, is a plot of density versus log exposure. To determine this curve, you need two sets of data: the measured photographic densities of a set of calibration spots and the log exposure values corresponding to these measurements. These data sets are determined by two tasks in the DTOI package. Task `spotlist` calculates the mean density of the calibration spots, each of which is a separate IRAF image or image section. Task `dematch` matches log exposure values to the measured density values. The log exposure values must be known a priori and will be read from a file. A database of log exposure values for the NOAO standard wedges is maintained in a system file. The objective is to fit a curve to these points, such that Log exposure = F(Density).

Task `hdfit` fits a characteristic curve to the density and log exposure values in preparation for transforming an image from density to intensity. It is possible to apply a transformation to the independent variable (density above fog) prior to the fit. Five functional forms of the curve are available as well, including linear or cubic spline, Legendre or Chebyshev polynomial, and a power series fit. The choice of transformation type, functional form, order of the fit and fog level can all be changed interactively. Also, data points can be deleted, added or edited interactively.

Once the HD curve has been defined, it is applied to a density image in task `hdtoi`. The transformation is accomplished by using a look-up table. On output, a new image is written; the input image is left intact. At the completion of task `hdtoi`, the goal has been achieved, a linear intensity image has been created.

Calibration data sets from several plates can be combined once a shift particular to each set has been removed. Different spot exposures define a series of HD curves which are parallel but mutually separated by arbitrary shifts in log exposure, produced by differing lamp intensities or exposure times. Task `hdshift` calculates and subtracts a zero point shift to bring several related HD curves into alignment.

Task `selftest` is a test task which investigates if any numerical errors were introduced during the density to intensity transformation. It also evaluates truncation errors produced when an output image with integer pixels, rather than reals, is written.

Suzanne Jacoby

THE PHASE DISPERSION MINIMIZATION PROGRAM

PDM is a new program in the IRAF `astutil` package. The algorithm used is called Phase Dispersion Minimization. PDM provides a means for users to find periodicities in time series data. An example data set is a light curve from a variable star (date vs. magnitude). The environment provided by the program includes numerous options for the batch or interactive user. The interactive user interface is a graphics cursor loop.

Generally, it is expected that the program will be used in interactive mode as the user will want to see plots of the spectrum and the compressed light curves as he does his analysis. A batch mode is provided for those cases where the data is very clean and many light curves need to be analyzed. The interactive mode will allow the user to fit and remove curves from the data or eliminate points. The batch mode will not allow this.

In interactive mode the user will always have a plot on the screen and will be in a graphics cursor loop. A set of cursor commands are available to allow the user to check and set parameters, to plot the data/spectrum/phase curves, and to perform various other functions.

In batch mode the spectrum is calculated, the minimum is located, the amplitude and epoch are calculated based on this period, and the compressed light curve is computed. The data plot, the spectrum plot, and the phase curve plot are saved in a specified metacode file. All other output information is saved to a log file.

Reference: Stellingwerf, R.F., 1978, "Period Determination by Phase Dispersion Minimization", The Astrophysical Journal, 224, pp. 953-960.

Dyer Lytle

UPDATE ON NEW IRAF PACKAGES - CCDRED AND APPHOT

Two new packages within IRAF, CCDRED, and APPHOT, have been under heavy development during the past year.

The `ccdred` package, written by Frank Valdes, is used for automated reductions of CCD data, including overscan, bias level, and dark frame subtraction, division by flat fields, etc. The package has been installed on the Sun workstations on both Kitt Peak and Cerro Tololo. Although this package is not included in the general distribution for IRAF version 2.5 it is now available upon request as an add-on to this release.

The `apphot` package, written by Lindsey Davis, is a general aperture photometry package for use in uncrowded fields. The package is now being tested in-house. There is no plan to publicly release this package until image cursor readback is available within IRAF. However, sites lacking other means for doing aperture photometry may request `apphot` as an add-on to version 2.5 with full understanding of the constraints of the package, i.e., that it is still in an early testing phase, that bugs may be present, that changes will probably be made to the package before it is formally released, and that the graphics terminal must be used in place of the image display for cursor input (some image cursor readback is available on the Sun).

Requests for these packages can be made either by noting your request on the order form or by contacting me directly.

Jeannette Barnes

FITS UPDATE

The FITS code has undergone some major revisions since the last release.

1. Multiple disk files can now be processed in a single execution of RFITS. The previous version of RFITS could handle multiple tape files but only a single disk file.
2. Read error recovery code has been added to the RFITS task. RFITS will now attempt to validate the input buffer and continue reading after it hits a bad data record. The previous version of RFITS would terminate and skip to the next file. The output images may contain regions of bad data but should have the correct dimensions.
3. The FITS code has been modified to handle FITS longblocks tapes, where the tape record size can be a multiple of the usual 2880 bytes. The maximum permitted blocking factor is 10. RFITS handles the variable record size transparently to the user. However a new blocking parameter has been added to WFITS. Caution should be used in exercising this option as only a few places currently support the FITS longblocks format.
4. RFITS will now issue a warning message if the IRAF default image user area allocated in memory is too small to hold all the FITS parameter cards being read by RFITS. The default user area is 8000 characters (or 4000 structure units) and the FITS card images are 81 characters long (a newline character is added by RFITS), allowing some 98 FITS parameters to be read in. RFITS will not allow partial FITS card images to be read into the user area. Other IRAF tasks that write header parameters into this area will also give warning messages when the area is full. The user can alter the size of the default user area by setting the environment variable *min_lenuserarea* to a sufficiently larger number of structure units, for example,

```
set min_lenuserarea=10000
```

Lindsey Davis

SIMPLE CL SCRIPTS

There are two important changes to the CL in V2.5 which affect *simple* CL scripts. Simple CL scripts are those which do not use the more formal *procedure-begin-end* syntax. Examples of these are the scripts which define packages, the scripts created by `mkscript`, and scripts created by simply typing commands in a file just as they would be interactively. These changes have been made to make writing scripts by you, the user, more consistent and useful.

The first change is to make all scripts except procedure scripts use the default CL syntax (as set by the CL parameter *lexmodes*). Briefly, the two types of syntax are *command mode* and *compute mode* and they differ in details of quoting strings and separating arguments by commas or whitespace. For virtually everyone the interactive CL syntax is command mode. The V2.5 change is that simple scripts, even those with curly braces, are consistently in the default syntax. Compute mode is used in several instances: when writing procedure scripts; when tasks are called using parenthesis, as in a subroutine call; and when a line begins with the characters "#{" (a line beginning with "#}" turns off compute mode). The rest of the line can be any comment.

This change does mean that scripts you have written in V2.3 IRAF may no longer work. You have two choices, you can rewrite the script or you may simply add a line at the beginning of the script beginning with "#{". You may notice that the package scripts in the IRAF packages are still in compute mode. This is a standard we have adopted primarily to make complex task declarations more readable. For those developing general packages you may want to follow this convention.

The other change which you will find useful is that other packages may now be loaded within a script. This is also described in the note on customizing your IRAF login. For scripts which require tasks from other packages that may not be normally loaded, the script can load these dependent packages. The packages loaded by a script will disappear when the task exits except in package scripts which end with `clbye` or if the command `keep` follows the loaded packages.

Frank Valdes

IRAF MAGNETIC TAPE I/O NEWS

During the past few months the IRAF magnetic tape i/o subsystem has undergone a series of tests to determine the best way to recover from tape parity errors. Test tapes with artificially introduced parity errors (holes) were made and test programs were written using high level i/o routines (*read(), write()*) and lower level i/o (*areadb(), awriteb(),await()*).

An ambiguity we encountered in these tests was where the tape is positioned by the host computer hardware/operating system following a read error. This seems to vary from computer to computer and to depend on the position of the parity error in the tape record. We found that in the majority of cases the tape is left positioned after the bad record. When the tape is left positioned before the bad record, the tape reader is faced with the possibility of an infinite read loop rereading the same bad record over and over forever.

The solution we came up with for this problem is to count the number of errors encountered reading any specific tape file and take actions based on this number. For the first few errors, the iraf magtape driver, *zfiomt.c*, returns ERR, then starts doing forward skip records to try to avoid the infinite loop described above. If a large number of read errors are encountered for a tape file then the driver returns EOF on the file.

This still gives some unpredictable results but is the best we can do, the host system is responsible for the rest.

The programmer, writing SPP code and calling the high level magtape i/o routines, can recover most of the data read when an error occurs in the following way. A read is done and, if an error occurs, the input buffer is validated (pointers are moved) and the read is redone. When validating, the programmer must input the number of chars to be validated. This number is either known or can be guessed based on the last successful read:

```
iferr (numchars = read (fd, buffer, SZ_BUF))  {  # if error on read
    call fseti (fd, F_VALIDATE, recsize)         # validate the buffer
    # Print an error message.
    numchars = read (fd, buffer, SZ_BUF)         # redo the read
}
```

If the low level i/o routines are used, the programmer is responsible for his own buffering:

```
call aread (input, buffer, maxchars, offset)     # start read
numchars = await (input)                         # read is done

if (numchars == ERR) {
    # Report read error.
    numchars = numexpected                       # reset numchars
}

bufferpointer = bufferpointer + numchar          # update pointer
```

The FITS tape reader has been modified to take advantage of the buffer validation described above and `mtexamine` has also been updated.

Dyer Lytle

MODIFICATIONS TO ONEDSPEC CALIBRATION FILES

The calibration data used by some of the tasks in the `twodspec, onedspec`, and many of the `imred` packages are kept in a directory called `onedstds` in `noao$lib/`. The contents of this directory are best summarized by its *README* file.

```
------------------------------------
```

This directory contains standard line lists and sensitivity calibration files for use in the ONEDSPEC, TWODSPEC, and the various IMRED packages.

LINE LISTS

ctiohenear.dat	-	HENEAR list used at CTIO
ctiohear.dat	-	HEAR list used at CTIO

henear.dat - Selected list of HENEAR lines to be used for KPNO IIDS/IRS reductions - this list is identical to that in the KPNO mountain reduction software for these instruments. Source of wavelengths: 1982 MIT Wavelength Tables Vol. 2. Multiplet numbers from Charlotte Moore's A Multiplet Table of Astrophysical Interest 1972.

idhenear.dat - Extended list of HENEAR lines for general use. Source of wavelengths: 1982 MIT Wavelength Tables Vol. 2. Multiplet numbers from Charlotte Moore's A Multiplet Table of Astrophysical Interest 1972.

thorium.dat - List of thorium-argon lines (3280 - 9048) courtesy of C. Pilachowski and Daryl Willmarth

EXTINCTION TABLES

ctioextinct.dat - CTIO extinction table (in Angstroms)

kpnoextinct.dat - KPNO extinction table (in Angstroms)

FLUX STANDARD DIRECTORIES

bstdscal - Directory of the brighter KPNO IRS standards (i.e. those with HR numbers) at 29 bandpasses

ctiocal - Directory of CTIO standards

iidscal - Directory of the KPNO IIDS standards (29 bandpasses)

irscal - Directory of the KPNO IRS standards at 78 bandpasses (note that in this directory the brighter standards have no values - the 'bstdscal' directory must be used for these standards at this time)

redcal - Directory of standard stars with flux data beyond 8370A. These stars are from the IRS or the IIDS directory but have data extending as far out into the red as the literature permits.

MISCELLANEOUS

ctio - Directory containing CTIO decker comb positions and neutral density filter curves (used with proto.ndprep).

- -

Major changes have been made to the format of the files containing information for flux calibrating your data. The old format was very restrictive making it difficult to add in data for other stars. This format has been made more flexible - each star's data is now a file and the stars are grouped by application into directories, as noted above.

The extinction files are now independent of the flux calibration files.

A new set of flux calibration data (`redcal`) has been added for stars with flux information past 8370A - previous KPNO lists did not go beyond this wavelength. The standard star data in this

directory are simply IIDS or IRS standards extended into the red if published data were available (Oke and Gunn, 1983, Ap. J. **266**, 713, and Oke, Ap. J. Suppl. Series No. 236, 1974, **27**, 21). Note that the Hayes and Latham corrections (Hayes and Latham, 1975, Ap. J. **197**, 593) were applied to the Oke standard fluxes beyond 8370A (see next paragraphs about IIDS and IRS standards).

The IIDS standards stars (`iidscal`) are from two sources: Oke, Ap. J. Suppl. Series No. 236, 1974, **27**, 21, and Stone, 1977, Ap. J. **218**, 767. The Oke standards have been corrected to the Hayes and Latham system (Hayes, private communication).

The sources for the IRS standard star data (`irscal`) as well as that for the bright list of standards (`bstdscal`) have been documented in the ***IRS Standard Star Manual***, Barnes and Hayes, 1982 (revised 1984).

The sources for the CTIO list of flux standards (`ctiocal`) are Stone and Baldwin, 1983, *MNRAS* **204**, 347, and Baldwin and Stone, 1984, *MNRAS* **206**, 241.

The data in the extinction files and the flux calibration directories can be listed using the task `noao.onedspec.onedutil.lcalib`. The data can also be plotted by piping the output of `lcalib` to the task `graph`. See the on-line help pages for `lcalib` for examples.

Jeannette Barnes
Frank Valdes

CUSTOMIZING HOST-SYSTEM AND IRAF LOGIN FILES

IRAF users may wish to customize their startup files, for example, to avoid having to specify the terminal type every time they log in from a different one, or to override the default location of bulk image or temporary file storage. In earlier releases, this was usually done in the `login.cl` file, which would be overwritten every time `mkiraf` was re-executed (as after a system update). In IRAF V2.5, a new convention has been adopted to preserve a user's special customization commands across updates. The file `login.cl` is retained (still created by `mkiraf`), for general system-wide initialization, but it now executes the file `loginuser.cl` if present in the user's login directory at login time.

Most entries in `loginuser.cl` will probably be portable IRAF commands, the same sort of thing many users are accustomed to editing in their `login.cl` files. However, other entries may depend in part on the host operating system, and are discussed separately below.

1. Examples of Portable loginuser.cl Entries

The default location for storing the bulk pixel file portion of an IRAF image is held in the environment variable `imdir`. On VMS systems this is usually `TEMPDISK`: [*username*], and in UNIX something like `/tmp2/iraf/`*username*/. To store the bulk pixel data instead in a subdirectory "pix" off your IRAF login directory, which is known to IRAF as `home$`, (note the trailing '/'), you can enter the following into your `loginuser.cl` file:

```
set imdir = home$pix/
```

If you always want a special set of packages loaded, and you tire of typing them manually every time, you might have:

```
noao
onedspec
mypkg
```

The last entry in the `loginuser.cl` file must be the command

```
keep
```

so that the modifications you have made will not be forgotten after the script has been executed.

2. Examples of Host-System Startup File Entries (and loginuser.cl)

Some users constantly log into IRAF from a variety of different terminals, and grow tired of remembering to '`stty` *terminal*'. Unfortunately there is no portable way to handle this for all operating systems (OS) and all terminals, because sometimes the IRAF name for a terminal is different from the OS's, and sometimes the OS doesn't recognize a terminal that IRAF supports. The IRAF environment variable mechanism may in some cases be used in conjunction with a host OS equivalent to pick up the name of the terminal from which you actually logged in. It is *not* a good idea simply to pass the OS's idea of a terminal name directly to IRAF for use in the `stty` command; IRAF may know the terminal by a different name, or the OS may be trying to tell you that you logged in over a network rather than what kind of terminal you are using. One should always have explicit branches for the list of terminals in use at a site, with a default in case there is no match. Examples are given for VMS and UNIX.

2.1. UNIX

UNIX shell and environment variables may be used to communicate the terminal type to IRAF. For example, the cshell variable `term` will often contain a useful indication of the terminal type. You can set the value of the cshell *setenv* variable `TERM` in your `.login` file, then test it in your `loginuser.cl` file. The following lines might be added to a `.login` file on, for example, an Ultrix VAXstation II/GPX running Xwindows (cshell example):

```
# Initialize Xwindows window manager if running on monitor.  Define
# an alias to use a nonstandard version of the terminal emulator.
switch ($term)
case xterm:
    setenv TERM xterm
    uwm &
    alias ixterm '/usr/iraf/local/xterm_kludge/xterm.init'
    breaksw
default:
    setenv TERM $term
    breaksw
endsw
```

Then, in your `loginuser.cl` file:

```
s1 = envget ("TERM")
if (s1 == "vt100")          # assume vt100 is Retrographics vt640
       stty vt640           # (UNIX doesn't recognize vt640)
else if (s1 == "vt240")
       stty vt240
else if (s1 == "sun")
       stty gterm
else                        # default; could use "stty (s1)" but IRAF
       stty vt100           # may not recognize UNIX name for terminal
```

In an example from a Sun workstation, we use the UNIX `tty` task to determine whether we are logging in from the console. In this case, we give the user 5 seconds to Control-C out of attempting to fire up `suntools`. The `.login` file:

```
if ($?WIDOW_ME == '1' || 'tty' == '/dev/console') then
    setenv TERM sun
else
    setenv TERM vt100
endif

(. . .)

# Now for starting IRAF automatically
if ("'tty'" == "/dev/console") then
    alias startIRAF "suntools; logout"
else
    alias startIRAF "cl ; logout"
endif
    echo -n "starting IRAF  (CTRL-C to return to UNIX)"
    sleep 5
    startIRAF
```

The `loginuser.cl` file given earlier would automatically set the IRAF terminal type correctly if you logged in from the workstation monitor. Other local terminal types could be added into the `.login` file as environment variables to cause `loginuser.cl` to set the terminal with `stty`.

2.2. VMS

The ability to determine what kind of terminal you logged in from on VMS is very dependent on the local environment and set of terminals. If it is possible on your system to determine your terminal type in your `login.com` file (see your system manager), you can define it as a VMS logical name, which will be accessible within IRAF as an environment variable. The local terminal port setup may determine whether it is possible to define the terminal type. For example, in your `login.com` file,

```
$ tname = f$getdvi(f$logical("tt"),"devnam")! get terminal name
$ if (tname .eqs. "_ttaO") then define TERM vt640! set env variable
```

Then your `loginuser.cl` file might look the same as in the preceding example from UNIX.

Steve Rooke

KEYWORD CHANGES TO NOAO FITS TAPES

NOAO is striving to adopt a standard FITS header for all of its instruments and image processing systems. As part of this process major upgrades were made to the FITS header keywords late last spring. (For more information on these changes, see *NOAO Newsletter*, No. 11, p. 37.) One of the keyword changes that could possibly affect IRAF reductions is the adoption of the keyword *exptime* for the time the detector actually sees light - this keyword in the past was *itime*.

Some IRAF tasks look for an integration time keyword, particularly many of the tasks in the `onedspec` package. In IRAF version 2.3, the only keywords recognized by the `onedspec` package for exposure time were *exposure* and *itime*. In IRAF version 2.5 the tasks will now recognize *exptime, itime,* and *exposure,* in that order. A quick way to see if your headers are readable by the `onedspec` package is to use the task `slist` and check the values of the parameters. If any of the information that is required by your reductions is set to INDEF, then you know that the `onedspec` package will not function correctly. You will need to edit the headers.

As an example, let us say that you are still running IRAF version 2.3 (but not for long, we hope). You have just brought some data back from the KPNO 4-meter telescope taken with the Cryogenic camera. You want to flux this data which requires the integration time. You execute `slist` on one of your reduced frames. For example,

```
slist trans006 "" 1+
```

This header listing shows the ITM variable as INDEF - this indicates that the `onedspec` package is not recognizing the integration time. Now execute

```
imhead trans006 1+
```

and you notice that indeed the keyword for the integration time is given as *exptime* and that there is no *itime* or *exposure* in the header. You can then use the `hedit` task to quickly modify the headers, adding the keyword *itime* with its value set to that of the keyword *exptime*. For example,

```
hedit tr* itime "(exptime)" add+ ver-
```

Jeannette Barnes

NOTES ON GRAPHICS DEVICES

Following are several items about graphics devices, as part of our continuing effort to provide a graphics interface that is both flexible and simple to install and use. References to further documents are mentioned herein.

1. Referring to Graphics Devices

There are several ways to direct output to specific graphics devices. In all cases, the IRAF graphics system needs to know the name of the desired device and must have the necessary device interface (see below). The system is distributed with default device names from our development system at NOAO.

When you run MKIRAF, you are queried for your terminal type (e.g. `vt640).` This then becomes an entry in your default login.cl file, e.g.:

```
stty vt640          # identify the terminal to be used
```

Interactive graphics terminals are also identified in two IRAF environment variables, `terminal` for the normal text mode, and `stdgraph` for vector graphics mode. When you switch to a different terminal, you need to tell IRAF about it with the `stty` command shown above (this command also sets both `terminal` and `stdgraph`, so do not set these environment variables yourself). This command can be issued from within the CL, as:

```
cl> stty selanar
```

If interactive graphics are not being displayed properly on your terminal, the first thing to check is that

the system has the correct device name:

```
cl> stty
```

See the two sections below for finding the supported device names on your system.

There are several ways to direct graphics output to hardcopy devices. Most IRAF graphics tasks have a `device` parameter, whose value may be specified directly, as in:

```
pl> graph blackbody.dat device=imagen
```

or indirectly, using the environment variable `stdplot:`

```
pl> graph blackbody.dat device=stdplot
```

In the latter case, your own environment is searched for the value of `stdplot`, which may be specially edited in your login.cl or loginuser.cl file or temporarily set as in

```
set stdplot = versatec
```

A system-wide default is also established at installation time.

When you are in cursor mode (see `help cursors)` and you wish to make a snapshot of the screen, the value of `stdplot` will be used unless you explicitly override it. Thus, when you use the = command (same as :.`snap`), output will go to whatever device is named in `stdplot`. If you wish to override temporarily the default device, you may specify it directly as in :.`snap imagen`.

Plots sent to certain devices may not appear immediately, but rather be stacked up in an internal queue, for greater efficiency. Such queues will drain automatically after an arbitrary number of plots has been requested (often eight at present), and always when you logout of the CL. To override the queue and force processing of a plot to begin immediately, execute the `gflush` command. This may be done either from within the CL, like any other task, or from within cursor mode, with the :.`gflush` command.

2. Vector Graphics Terminals and the Graphcap Entry

We get a number of questions about interactive vector graphics terminals. Virtually all vector graphics terminals that have a serial interface to the host computer may be installed in IRAF through an entry in the graphics configuration file `dev$graphcap`. You may page that file to search for an entry for a given terminal. See also the sections covering `termcap` and `graphcap` entries in your site's *IRAF Installation Guide.*

If there is no specific entry for a particular terminal, the terminal may have an emulation mode for which there is an entry. For example, if your terminal has a Tektronics 4014 emulation mode, you may try:

```
cl> stty 4014
```

However, even if this works, it will probably not be what you want, because it will emulate the Tektronics terminal faithfully, including no erasing or text scrolling. So you may want to try some of the other Tektronix emulators. Look for the device names for those entries that point, directly or indirectly, to one of the Tektronics entries with `tc=`. For example, the `vt640, hds, 4010, 4014, wyse,` and `414a` entries all have `tc=4012.` Even if you don't know anything about graphics protocols, you can always use trial and error to see if any of these entries work.

If no existing device entry is suitable, it is necessary to create a new one. Experienced programmers can examine the *Graphics I/O Design* manual by Doug Tody, revised April 1987, along with existing `graphcap` entries, to learn how to do this. A hardcopy of this manual is included in the

IRAF System Handbook, Volume 3B, while the source document is in `gio$doc/gio.hlp`. If you do not have access to a programmer, but you want to give it a try anyway, we will be happy to help to the extent we can (sometimes you just have to see the terminal physically). In some circumstances we have been able to share shipping costs for the loan of a terminal for a few days.

3. SGI Device Installation

We also get a lot of questions about installing hardcopy plotter devices within IRAF. Although it is well outside the scope of this Newsletter to cover the subject fully, a few tips are in order. Most hardcopy plotters are installed within IRAF using our *Simple Graphics Interface (SGI)*. Device installation is discussed in the *IRAF Installation Guide* for your host operating system, and is described more fully in the paper *The IRAF Simple Graphics Interface (SGI)*, by Doug Tody, August 1986.

The first thing to do when you acquire a new plotter is to see if your system already includes an interface for it. Simply attempting to plot to the device using your local system name for it is not sufficient, because it may be installed under another name. You can start by paging the IRAF graphics device configuration file, `dev$graphcap`, to search for a device name that sounds like your plotter's. In that file the SGI device entries are probably grouped together.

Simply finding an entry in `dev$graphcap` is not necessarily enough. Many SGI devices also require *translators*, or device drivers. A programmer can figure out whether a translator is required by looking at the `DD` entry for the device in the `graphcap` file (note that you may have to follow a chain of `tc=name` entries, where "name" is that of another device entry elsewhere in the file). Each installation comes with some set of SGI translators, which, being device and operating-system specific, live outside the main IRAF system in the *Host System Interface*.

If a translator is required and available, its executable is probably in the `hlib` directory, beginning with `sgi` -- e.g. on a VMS system the SGI translator for the Imagen using the Impress language is in `[iraf.vms.hlib]sgi2vimp.exe`. The source files for the translators are in the `host$gdev/sgidev` directory, e.g. `[iraf.vms.gdev.sgidev]sgi2vimp.for`. If special installation steps are required to install a translator on your host machine, the instructions will be found in the *IRAF Installation Guide*.

If a translator does not yet exist for your device on your operating system, contact the *IRAF Hotline* to see about obtaining or modifying one.

Steve Rooke

NOTE ABOUT VMS SHARED LIBRARIES

Load-and-go tapes of VMS/IRAF are configured to use shared libraries. This is discussed in the VMS/IRAF Installation Guide. If you do not modify the system (recompile any part of the IRAF system code), you should experience no problems providing you followed the installation steps in the Guide, and provided you have only one version of IRAF on your VAX.

If for any reason you do rebuild part of the IRAF system, several warnings are in order. After recompiling any routine in the libraries hlib$libos.a, lib$libsys.a, lib$libex.a, and lib$libvops.a, you must rebuild the shared library. See the instructions in the Installation Guide. Then, *before* relinking anything, make certain you run the VMS INSTALL utility and use its DELETE option to remove all existing installed IRAF images including s_iraf.exe. Only then is it safe to reinstall the shared library and other tasks. Note that it is not enough simply to execute hlib&install.com; that command file does not at present delete existing installed images before reinstalling them. Then make certain you relink *all* IRAF applications plus any of your own before trying to run anything.

Contact us for advice if you have any problems related to the shared library, and also for suggestions about debugging tasks linked with it. If you experience problems and cannot reach us right away, you can simply de-install the shared library and relink all executables without it (see the Installation Guide).

Steve Rooke

IRAF NETWORKING WITH DECNET

IRAF networking between hosts that both run IRAF is functional, though it is still evolving. In particular, many sites now require some sort of network interface to gain access from workstations to plotters on remote nodes. The IRAF installation tapes are configured for the TCP/IP network protocol; if you have DECNET instead, you will need to edit some files, perform a bootstrap, and relink all the applications. Since we are still working on the DECNET implementation, we do not provide a guide to configuring IRAF for DECNET, but rather request that sites contact us for current instructions.

Steve Rooke

IRAF BENCHMARKS

The IRAF Benchmarks paper has been extensively revised, extended, and typeset, and benchmarks for a number of new systems are now available. The summary sheet for the systems thus far benchmarked (one has to do an IRAF port first) is reproduced in this newsletter. We expect to add benchmarks for a number of interesting new systems over the course of the next six months or so.

The standard benchmarking caveats apply to these IRAF benchmarks just as to any other benchmark suite. Many of the benchmarks shown are quite complex and therefore difficult to interpret. Since these are IRAF benchmarks the performance of IRAF is being tested as well as that of the host system, so one must be careful when trying to make comparisons between different host systems. A different set of benchmarks might lead one to draw a completely different set of conclusions. Sometimes a system feature (e.g., the UNIX global buffer cache) can have a major impact on a benchmark, but this may or may not imply a comparable advantage or disadvantage when using the system (it depends upon the application). Often the specific hardware configuration of the system used for the benchmarks will significantly affect the results, and this can reflect unfairly on a vendor even though it may be the fault of the user if the hardware configuration is inefficient.

The systems being tested are continually changing, often making the benchmarks obsolete by the time they appear in print. The benchmarks and their meaning and interpretation is discussed in detail in the *IRAF Benchmarks* paper.

Doug Tody
Suzanne Jacoby

NEW IRAF LOGO

You will note that this issue of the IRAF Newsletter is sporting the IRAF project's new logo!

Earlier this year an NOAO-wide contest was held in search of an IRAF logo. The response was overwhelming! There were more than 60 entries, and the judges (whose names shall forever remain anonymous!) had a very difficult but exciting challenge. The winning logo was submitted by Clark Enterline in the NOAO Procurement Office, with second and third places going to Nigel Sharp and Maggie Lawrence, respectively. All the logos are on display in the central corridor of the NOAO building - for those of you who would like to see the artistic talents of our NOAO folk the next time you are in Tucson.

Jeannette Barnes

ELECTRONIC COMMUNICATION WITH THE IRAF PROJECT

There are two electronic ways to send gripes, kudos, comments and questions to the IRAF project. First, you can send electronic mail via several computer networks. On the Internet (ARPAnet, MILnet, NSFnet, etc.), send mail to the following address (case is not significant).

```
iraf@noao.arizona.edu
```

On the BITnet, send mail to the same address through the Wisconsin gateway (consult a local network guru for the details at your site). On the SPAN/HEPnet network (DECnet), send mail to

```
draco::iraf
```

If DRACO is not in your host table, substitute 5356 for DRACO. Finally, on the UUCP/Usenet network, send mail to

```
ihnp4!noao!iraf     or     uunet!noao.arizona.edu!iraf
```

If your computer is cut off from the world, or if the preceding paragraphs are just gibberish to you, you and your modem can call one of our systems directly and leave mail. Our modem's phone number is (602) 325-9259 and it will answer at either 1200 bps or 300 bps. Set your system for 8 bits, no parity and one stop bit. Once your modem connects to ours, hit CR a couple of times and you will see some interesting introductory text and (after a short wait) be put in the Software Tools mail program called 'msg.' You will see a '<—' prompt. Type the '%' key (no CR) for some immediate help. There may be some messages in the mailbox for your perusal; type an immediate CR at the prompt to see them in sequence. PLEASE do not delete the messages!

You may use the 's' command to send mail to iraf. To do so, follow this recipe:

1. Type 's' at the '<—' prompt (no CR) and type a space when prompted.
2. Enter 'iraf' at the 'To:' prompt (from now on, you need CRs after your entries); then enter CR to the subsequent 'To:', 'cc:' and 'Bcc:' prompts.
3. Type in an appropriate line following the 'Subject:' prompt.
4. Answer 'n' to the 'Do you want a return receipt? [y/n]' question.
5. After the 'If you need help, type h' message, type 'a' to the 'sm>' prompt (don't forget the CR!).
6. Now enter your message on the subsequent lines. Be sure to identify who you are and how we may get in touch with you in the body of your message. Terminate your message with a lone '.' followed by a CR on a line.
7. Enter a 's' following the 'sm>' prompt and answer 'y' to the 'Are you sure? [y/n]' question. You will see a message 'Submitting mail to MAILER'.
8. Finally, enter 'q' (no CR) to the '<—' prompt. You will be logged off from VMS and the phone line will be hung up.

A real live person whom you can consult on electronic communication with the IRAF project is Steve Grandi at (602) 325-9228. If you are reading this memo more than a few months after it was written, please check with us before using any of the recipes: things change quickly!

Steve Grandi

The IRAF NEWSLETTER is published by the Central Computer Services, National Optical Astronomy Observatories, P.O. Box 26732, Tucson, AZ 85726. Editors: Jeannette Barnes, Doug Tody