IRAF NEWSLETTER

June 1989 Number 7

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 HOTLINE SERVICES

telephone:	(602) 323-4160
Internet:	iraf@noao.edu
SPAN:	noao::iraf or 5355::iraf
BITnet:	iraf@noao.edu (through a gateway)
UUCP/Usenet:	{arizona,decvax,ncar}!noao!iraf
	uunet!noao.edu!iraf

The IRAF NEWSLETTER is published three times a year (February, June, and October) by the Central Computer Services, National Optical Astronomy Observatories, P. O. Box 26732, Tucson, AZ 85726. Editors: Jeannette Barnes, Doug Tody

System News

The major activity for the IRAF group during the period covered by this newsletter was preparation for the IRAF V2.8 release, which was frozen in late June and which we started shipping in early July. This was a major release on all supported systems, incorporating many additions and enhancements, a revised distribution and installation format, support for new systems such as the DECstation and Convex, support for layered software such as the just released STScI STSDAS system, and more. See the Revisions Summary and other articles in this newsletter for further information.

In the area of science applications, both the IRAF DAOPHOT (digital stellar photometry) and RV (radial velocity analysis) packages have reached the user test stage, with beta test releases expected by the end of the year. V2.8 incorporates many additions to the existing science packages, including a new Echelle reduction package (many people will already have seen this) and a new package for reducing multispectral data.

An IRAF User's Committee, formed of representatives from all segments of the ground based astronomy and space sciences communities using IRAF, has been formed and will meet for the first time in September of this year. An IRAF Technical Working Group representing the major groups using IRAF for software development is also in the process of being formed.

As we go to press, we have been informed that we have received a three year grant from the NASA space astrophysics division to accelerate IRAF core systems development. This reflects the increasing use of IRAF by the space sciences community, but will benefit the ground based community as well. We plan to use part of this money to hire someone to help develop and maintain the IRAF system software; a job announcement appears elsewhere in this newsletter. If you know of someone who might be interested in this position please ask them to contact us!

Doug Tody

Things to Watch Out For with IRAF V2.8

IRAF V2.8 has been frozen and in distribution for a couple of months. As is always the case with any major release, a number of minor problems have been identified since the system was frozen. Foreknowledge of such problems can avoid considerable aggravation, so the most notable of the pitfalls associated with installing and using V2.8 are summarized here.

Linking IMFORT Programs

When linking IMFORT programs from within the IRAF environment using FC on a Sun-3, be sure to specify the architecture (e.g., "f68881" or "ffpa") so that it matches that of the IRAF system being run. For example, if you want to develop programs compiled for the f68881 floating point option, define IRAFARCH and FLOAT_OPTION in your SunOS environment as "f68881" and you won't have any problems. If a mismatch occurs, e.g., if Fortran modules compiled for f68881 are linked with the ffpa IRAF libraries, then your program will appear to link correctly but may crash mysteriously at runtime with no clue as to what is going on. Prior to the introduction of SunOS-4, this was much less of a problem as an error message such as "unresolved symbol ffpa_used" would be issued at link time, but this no longer occurs with SunOS-4 due to the introduction of shared libraries in SunOS.

Forwarding of IRAF mail

The various UNIX/IRAF systems as distributed may include a `.forward` file in `iraf$local` which will forward mail sent to the iraf account to `iraf@noao.edu`. While this does make a certain amount of sense, it has caught a number of people by surprise and you may want to delete the file.

IRAFARCH=generic

The `.login` file in the distributed Sun/IRAF system includes a few statements at the end of the file which will set the IRAFARCH environment variable to match the architecture for which the core system is currently configured, which is `generic` in the distributed system. This is appropriate for core systems software development, but will cause a warning message stating that the bin.generic version of the CL cannot be executed (since there is no such file) when the CL is run from the IRAF account, e.g., as part of the test procedure. This is normally harmless since a different architecture CL will be run, and only the IRAF account is affected, but you may wish to edit the `.login` file to change the way IRAFARCH is defined.

Complex Networked Installations

A discussion of the general approach for installing IRAF on a complex client/server type network was inadvertently omitted from the installation and site manager's guides. If such a network contains diskless nodes the system files for the clients will reside in file system partitions on the server. If, as part of the normal installation procedure, one runs the INSTALL script on each client as root, the script will fail to make the necessary links since root is not normally allowed to modify files on a remote machine. The solution is to run the INSTALL script *on the server* for each client, specifying the client architecture (which may differ from that of the server) with the "-m" switch to INSTALL, and specifying the `local/bin` and similar pathnames as those for the client partition. INSTALL must then be run a second time on each client to make the IMTOOL device entries in `/dev`. While this procedure is straightforward it is difficult to explain and each system is different, so anyone who runs into trouble with this one will probably have to contact the HOTLINE for help.

SunOS-3 Bugs

The V2.8 IRAF distribution for SunOS-3 (*not* SunOS-4) contains one file in APEXTRACT which must be recompiled without optimization to avoid a Sun Fortran optimizer bug, and there are a couple of problems with the XC program (SPP/VOS compiler) distributed with the system. Detailed notes describing these problems and their workarounds are included with each distribution.

IMTOOL on a SPARCstation 1

We were unable to test V2.8 IRAF on a SPARCstation (desktop unit) until after the system had been frozen. The system works fine except for a problem with the cursor in the IMTOOL window, evidently caused by a bug in the SunOS kernel or frame buffer driver on this system (the problem does not show up on any other Sun system). There is a workaround which will avoid the problem until we have time to find a way to hack the software to avoid the bug. See the accompanying article in this newsletter for further information.

SunOS 4.0.3, VMS 5

Sites upgrading to either of these popular operating system versions should know that while the distributed IRAF V2.8 binaries appear to run fine on both of these systems, we have not yet rebuilt IRAF on either system. Rebooting or recompiling IRAF on either of these systems is inadvisable until we have had a chance to check things out. Ordinary applications software development using the distributed system should not present any problems.

More detailed information on some of these items is given in the system buglog, in auxiliary documentation included in each IRAF distribution, or in the other articles in this newsletter. Additional information is available via electronic mail addressed to the iraf account at NOAO.

Doug Tody

Job Announcement

The IRAF project has an opening for a systems programmer. Primary duties will include projects involving the development and maintenance of the IRAF core system. Actual project assignments will depend upon the interests and abilities of the person selected for the job. This is an opportunity to make a significant contribution to the development of the IRAF system, working with computers ranging from personal computers to supercomputers, at sites throughout the US and the world.

The minimum requirements for this job are several years programming experience demonstrating an aptitude for working with and developing complex systems software. Knowledge of the technology employed in modern scientific workstations, and experience with modern software engineering methodology is highly desirable. Experience with UNIX/C or IRAF software development would be a plus. Although this is primarily a systems software position, an interest in astronomical or scientific software development is considered highly desirable for anyone involved with the IRAF project.

The IRAF project provides one of the most dynamic and stimulating environments for software development available in the astronomical or space sciences communities today. NOAO offers an excellent compensation and benefits package including 24 days annual vacation. Please send resume and salary history, in confidence, to

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

Doug Tody
Steve Ridgway

Beta Release of Convex/IRAF V2.8 Now Available

Convex/IRAF version 2.8 is now available for distribution as a beta test release. IRAF version 2.8 was installed and tested on the Convex C1 computer (Convex Release 7.0, Vectorizing FORTRAN Version 5.0) in Socorro at the National Radio Astronomy Observatory in early August. The initial installation was done remotely via the internet, with the final system testing being done on site over a long weekend.

The initial port to the Convex had been done in October 1987 using the Convex machine at the Convex headquarters in Greenbelt, MD, using a much earlier version of the FORTRAN compiler. This earlier version of the compiler had some serious bugs, but things seem much better now. With the current compiler only one file in the whole IRAF system failed to compile due to a compiler bug, and thus far no runtime compiler problems have been encountered.

The Convex system has been configured for two architectures, NATIVE and IEEE, corresponding to the two floating point hardware formats available for the Convex. The IEEE format provides file formats compatible with Sun3's and Sun4's, while the NATIVE format is compatible with a VAX. Binaries for both architectures are included in the beta V2.8 Convex/IRAF distribution. Both sets of binaries were compiled for one cpu, with automatic vectorization enabled for the science applications code.

Convex/IRAF uses the Convex tape allocation facilities, TPALLOC and TPDEALLOC, to perform the actual IRAF allocate/deallocate. This was tested for both remote and direct magtape access, between two Convexes or between a Convex and a Sun.

Most of our tests were performed using a Sun workstation for graphics and image display, with IRAF running on the Convex. IRAF could also be run on the Sun and used to access images or other files on the Convex. As a final test we tried running IRAF on the Convex from a Sun workstation in Tucson. This worked fine, providing a usable system although the relatively low network bandwidth was noticeable (it would take about a minute to display an image, with the display program in Socorro writing to the IMTOOL window on the workstation in Tucson).

To date 8 sites have requested and received Convex/IRAF.

We would like to express our thanks to the folks at the VLA for their generous hospitality in allowing us to use their facilities for this IRAF port. We would like to thank Bob Payne, in particular, for making all the necessary arrangements and for spending a weekend away from his family so he could give instant help to us while we were testing at the site.

Doug Tody
Jeannette Barnes

IRAF Ported to the DECstation 3100

IRAF version 2.8 was ported to the new DECstation 3100 (MIPS RISC cpu, Ultrix 3.0) in mid-July. This early version of the system is now available for distribution and testing as an alpha port. Although the basic port is complete users should be aware that much testing still needs to be done, and there are already some known if apparently minor problems. In particular, the IEEE exception handling is not working correctly, and the floating point numerical results of some programs (the FITS and IMHISTOGRAM programs thus far, but there are probably others) differ noticeably from other systems we have worked with.

Some vector graphics support is provided for the DECstation itself. The DECterm vt3xx emulator supports ReGIS graphics for output, but due to a bug in Ultrix 3.0 it does not support cursor readback (this should be fixed in Ultrix 3.1). The XTERM terminal emulator may be used successfully with IRAF using the two termcap interfaces, xtermr and xtermjh, provided with IRAF version 2.8. Work on image display for the DECstation, using DECwindows and X11, is still in progress.

We would like to thank our local DEC office for the loan of the DECstation 3100 so that we could provide the IRAF community with this alpha port.

Doug Tody
Steve Rooke

Status Report on the Apollo/IRAF Port

We received our donated Apollo DN-3500 with FPA the last week of May. The system is now up and running and the basic IRAF port has been completed, but more work remains to be done before the first release of Apollo/IRAF is ready for export. Due to a bug in the Apollo software we have not yet been able to prepare and test the FPA version of the system. Process startup is slower than we would like and we are currently investigating the use of shared libraries to try to improve performance. The system as originally delivered did not include X11 (the tape has just arrived as we go to press), and the current Apollo version of XTERM (the X graphics terminal emulator) has vector graphics disabled, so additional work remains to be done before the system is usable for applications involving graphics and image processing. System testing and science software verification remains to be done.

Anyone interested in obtaining IRAF for an Apollo should contact us for up to date information on the progress of this port.

Steve Rooke

IRAF on the SPARCstation

We have been successfully running SunOS4/IRAF version 2.8 on a SPARCstation running SunOS4.0.3c. No problems have been encountered within any of the IRAF and NOAO packages. A minor problem has been encountered in IMTOOL on the SPARCstation with IRAF version 2.8 in that the IMTOOL image cursor leaves "scratches" on the IMTOOL window (and sometimes outside it) as the cursor is moved about in the window. Sites running IRAF on a SPARCstation are encouraged to contact the IRAF HOTLINE so that their names can be added to the distribution list for the software patch when it becomes available. Meanwhile, as a workaround, one can erase the scratches by refreshing the screen, or avoid them entirely by executing `=imcur` in the CL to enter image cursor readback mode before moving the mouse about in the IMTOOL window.

Jeannette Barnes

IRAF Version 2.8 Distribution in Progress

We started distributing IRAF version 2.8 to sites in early July. To date, we have shipped the VMS, SunOS3, SunOS4, BSD, Convex, and DECstation 3100 versions of the system. AOSVS/IRAF, currently being prepared by Steward Observatory, is expected to ship soon. Ultrix/IRAF (supported only for Ultrix 3) is on hold until we have a working version of SAOIMAGE for X11 (SAOIMAGE is the enhanced X11 version of the old X10 program XIMAGE). The remaining distributions, e.g., for the Alliant and HP, will ship as soon as the system updates have been completed.

Jeannette Barnes

IRAF Supported SGI Graphics Devices

The following devices are supported SGI graphics devices within IRAF V2.8. SGI translators for those devices marked with an asterisk [∗] were not distributed with V2.8, but are available by contacting the IRAF HOTLINE (602-323-4160).

UNIX/IRAF:

PostScript devices such as an Apple LaserWriter
Impress language (Imagens)
Printronix plotter
QMS vector graphics mode (Talaris Lasergrafix, QUIC command mode)
Versatec
HP Graphics Language (e.g., HP7550A pen plotter) [∗]
HP LaserJet II [∗]

VMS/IRAF:

PostScript devices such as an Apple LaserWriter
Impress language (Imagens)
Printronix plotter
QMS vector graphics mode (Talaris Lasergrafix, QUIC command mode)
Versatec
HP Graphics Language (e.g., HP7550A pen plotter)
HP LaserJet II
Calcomp pen plotters
Trilog plotter
DEC LN03+ (in TEK mode)
DEC LN03R ScriptPrinter (DEC’s PostScript printer) [∗]

Suzanne Jacoby

Capital Letters in VMS/IRAF V2.8

There is a bug in VMS/IRAF V2.8 such that capital letters should not be used to name IRAF images. Image names containing capital letters are not interpreted properly by the IMDELETE and IMRENAME tasks, as seen in the following commands:

```
cl> imcopy dev$pix XYZ
cl> dir
XYZ.imh
cl> imdelete XYZ
Warning: Cannot access image (XYZ)
cl> imrename XYZ xyz
Warning: cannot rename ’XYZ’ --> ’xyz’
```

Additionally, image names containing a capital J show other incorrect behavior:

```
cl> imcopy dev$pix JUNK
cl> dir
jUNK.imh
cl> imdelete jUNK
Warning: Cannot access image (jUNK)
cl> imdelete JUNK
Warning: Cannot access image (JUNK)
```

This will be fixed in a subsequent IRAF release, but for now it is best to avoid the use of capital letters when naming IRAF images in VMS/IRAF.

Suzanne Jacoby

A New Task Called TVMARK

A new task TVMARK has been written to mark objects on the image display by writing directly into the image display frame buffer. TVMARK is useful for identifying objects in a list, for example objects detected by the DAOFIND task in the APPHOT package, for making finder chart images, and for displaying regions of interest around objects, for example the apertures and sky annulus used for photometry.

TVMARK can be run interactively or in batch mode. In interactive mode the objects to be marked may be selected interactively by using the image cursor or chosen from a text file based on position in that file. New objects can be appended to an already existing coordinate list. All the parameters including frame number, coordinate file, and the mark type, color and mark size can be changed interactively. In non-interactive mode all the objects in the coordinate list are marked on the display with the default mark type. At any point in the marking process a snapshot can be taken of the current contents of the frame buffer and saved as an IRAF image.

The currently supported marks are "none", "point", "plus", "cross", "circle", a list of concentric circles, a list of concentric rectangles, and line segments. Options exist to number or label the marked objects. A simple font is provided for drawing text into the frame buffer as well.

TVMARK can draw on any device currently supported by the IRAF DISPLAY task, although some features like colored marks are only available on the Sun IMTOOL display. TVMARK is included in the standard IRAF V2.8 distribution.

Lindsey Davis

IMEXAMINE - A New Task for Examining Images

A new task has been added to the NOAO.PROTO package in V2.8 for interactively examining images. Images are examined using the image display, various types of vector graphics plots, and text output. Commands are given using the image display cursor and/or graphics cursor. This task brings together many of the features of the IRAF image display and graphics facilities with some simple image analysis capabilities.

Although the task may be used interactively without an image display (or even a graphics terminal), or noninteractively via cursor list files, its most common and powerful usage is with an image display and graphics terminal, e.g., a workstation running IMTOOL or SAOIMAGE, or a graphics terminal and hardware image display (for example a VT640 and IIS).

The main argument to the task is an optional list of images. If an image list is given one moves about through the list, with the task displaying the images upon demand as one might page through a list of files. If no image list is given then the images already loaded into the image display are examined. Additional images may be displayed once the task is entered. The command options include a variety of graphs and text output based on the part of the image pointed to by the image cursor.

The graphics output includes contour maps, surface plots, pixel value versus position along columns, lines, and arbitrary vectors, histograms, and radial profiles about a point with gaussian fit overplotted and photometry and shape information printed on the status line. All these graphs have parameters to select sizes, labels, averaging, and more.

The text output includes position and intensity at a point, a square array of pixel values, statistics in a box, photometry and gaussian profile fits, and radius and angle from a given origin. This output may be toggled to go to a log file.

The radial profile and photometry features provide simple and quick estimates of position, size, ellipticity, flux, and background. The background options include no background, median, and background surface fit. The photometry is a simple whole-pixel aperture flux within a user specified radius. The centering uses marginal distributions. The ellipticity is based on radial moments. The circular gaussian fit is constrained to the previously determined center and background and provides a peak and sigma.

The task is powerful and fun to use. As implied by being in the PROTO package, this is a prototype task and there may be additional features added. In combination with PROTO.IMEDIT and PROTO.TVMARK V2.8 provides a great improvement in your abilities to interact with your images visually.

Frank Valdes
Doug Tody

IMEDIT: New Image Examination and Pixel Editing Task

A new task for interactively examining images and editing pixel values using the image cursor has been written. This prototype task is PROTO.IMEDIT. Some features of the new task are:

- Load, zoom, and roam of the image display interactively.
- Surface plots of image regions at the cursor position.
- Statistics and pixel values at the cursor position.
- Replacement of individual pixels, circles, squares, lines, and rectangles.
- Replacement algorithms include:
    - constant
    - surface fit of background annulus plus noise
    - column or line linear interpolation plus noise
    - one region copied to another by replacement or addition.
- Before and after surface plots of the changes.
- Previous changes can be undone.
- Replacement size and background buffer and width dimensions may be adjusted.
- Background may be fit by arbitrary polynomial surface.
- All parameters may be modified interactively by colon commands.
- Input from a cursor, list of cursor commands, x-y file, or fixpix type file.
  The x-y files may be obtained from various sources such as the photometry packages or the COSMICRAYS task.
- Task may be run noninteractively using one of the inputs above.
- Record of all commands modifying the image. This record can be used as input
  to reproduce the modifications on other images.

This new task is included in the standard IRAF V2.8 distribution.

Frank Valdes

Specifying Pixel Directories with IMFORT

IMFORT has been modified in IRAF version 2.8 to permit specification of the pixel file directory by the calling program. The modifications are upwards compatible, i.e., existing programs linked with the new interface will still create pixel files in the same directory as the header file, with "HDR$" in the image header.

The Fortran programmer may set or query the pixel file directory using the following routines:

```
call imsdir (dir)             (set pixel directory pathname)
call imgdir (dir)             (get pixel directory pathname)
```

where `dir` is a Fortran character variable. The value should be either "HDR$" (the default), a variant such as "HDR$pixels/", or a concatenatable host directory pathname (i.e., trailing / required for unix). Once set, the pixel directory will be used for all subsequent image create or rename operations by the calling process.

If desired the default pixel directory may be specified in the host environment as `imdir` or `IMDIR` before running the program. IMFORT will check the host environment for this environment variable then use "HDR$" as the default if no host definition is found. For example, in the UNIX csh environment, use "`setenv imdir` */path/*". In the VMS environment, use either "`define/job imdir` *disk*:[*dir.subdir*]", or if the VMS/IRAF system manager has set the global logical name `IRAFIMDIR` for all users, it will be used automatically unless you override it with your own (i.e., whichever of `IRAFIMDIR` or `IMDIR` has the highest VMS logical name precedence will be used).

Note that although this is similar to setting the value of `imdir` in the IRAF environment, IMFORT programs are not part of the IRAF environment and are not affected by changes to the IRAF `imdir`. Also, since IMFORT is a host level facility and IRAF networking is not supported, the network prefix (e.g., "node!") is omitted, and since IMFORT programs are not necessarily used in conjunction with IRAF, the ".." files are not used to protect against image deletion. Also, if imdir is set to a nonexistent subdirectory, IMFORT will not automatically create the subdirectory, unlike in IRAF.

UNIX example:

```
call imsdir ('/tmp3/pixels/')
call imcrea (image1, axlen, naxis, dtype, ier)
call imcrea (image2, axlen, naxis, dtype, ier)
```

Or, using environment variable imdir, (could be in `.login`):

```
% setenv imdir /tmp3/pixels/
```

VMS example:

```
call imsdir ('scr$0:[mydir.pixels]')
call imcrea (image1, axlen, naxis, dtype, ier)
call imcrea (image2, axlen, naxis, dtype, ier)
```

Or, to use the VMS logical name method (could be in `login.com`):

```
$ define/job imdir scr$0:[mydir.pix]
```

The job logical name is used in case the IMFORT program is interfaced as a subprocess; otherwise "/job" is not needed.

Doug Tody

APPHOT Update

Many of the new features available in the APPHOT package running under IRAF version 2.7 were discussed in an earlier issue of the IRAF Newsletter (October 1988, Number 5). Some additional features have been added with the release of IRAF Version 2.8 and these are discussed below.

A new parameter "scale" has been added to the APPHOT package. Scale is defined as the image scale in world coordinates per pixel, for example arc seconds per pixel. If scale = 1.0, the default, all the distance dependent parameters including the full width half maximum of the point spread function "fwhmpsf", the centering box width "cbox", the inner sky annulus "annulus", the width of the sky annulus "dannulus", and the aperture radii "apert" are in pixels, otherwise these quantities are in the same units as the world coordinates. All radial profile plots now display both pixel and world coordinate units. Users should note that the "fwhmpsf" parameter has been decoupled from the definition of all the distance dependent parameters. These parameters can still be defined in units of the "fwhmpsf" by defining "scale" in units of fwhmpsf per pixel and setting "fwhmpsf" = 1.0.

Two new switches "verify" and "verbose" have been added to all the APPHOT tasks. In non-interactive mode the verify switch prompts the user to confirm or change the parameter values critical to the task. In interactive mode the 'v' keystroke command will perform the same action. The verify option should be turned off if tasks are submitted to run in the background. The verbose mode permits the user to see a short summary of the results for each object when the task is running in non-interactive mode.

Lindsey Davis

RV Package Available for Testing

The Radial Velocity Analysis package is now available for testing on a user-beware basis. While not yet fully complete, the cross-correlation tasks and support software are in place and have produced reliable results in testing at NOAO. The package is currently able to cross correlate both one and two dimensional spectra (both longslit data as well as multispec and echelle format data), and a Fourier Quotient task is available for galaxy work. The package is currently configured as an external package, thus requiring the layered software enhancements found in IRAF V2.8, but it can be distributed to be compatible with older versions of IRAF.

Planned work includes the implementation of a Fourier Difference task, a task to fit emission line profiles for velocity analysis, a task to remove telluric (or other artificial) features, and a task to select fields from the output database, as well as enhancements to the FFT plotting capabilities and input data formats. Suggestions for new tasks or improvements to existing tasks are welcomed. The package has only been tested on a limited set of data so problems are expected, but most problems encountered so far have been minor and a patch or workaround can be supplied relatively quickly.

For more information on this package or how it may be obtained, please contact Mike Fitzpatrick at (602) 325-9387 or through e-mail at fitz@noao.edu or 5355::fitz.

Mike Fitzpatrick

The IRAF DAOPHOT Package

Considerable work has been done on the IRAF DAOPHOT package delivered to NOAO by Dennis Crabtree in February. A month was spent in Chile re-engineering portions of the package to integrate it better into the IRAF environment, tracking down bugs, and getting input from the CTIO scientific staff. The major change to the package at that time was adding the capability to support both text and ST table format for both input and output files. The package task will now automatically sense which type of file is being supplied as input and the user can select the type of the output file by setting a switch.

On return to NOAO considerable effort was spent validating the numerical accuracy of IRAF DAOPHOT by comparing its results with those of Peter Stetson's version running on a VAX under VMS, working in collaboration with Phil Massey. Several problems both major and minor surfaced at that time but have been resolved. Finally the 'allstar' task and the variable psf capability which were not supplied with the original version from DAO were written and tested. Work is now continuing on improving the user interface and on writing some auxiliary list processing programs.

Collaboration with Dennis Crabtree, now at STScI, is continuing in the area of supplying tools to allow the user to interactively examine and edit the output datafiles.

Plans for a beta release of the package will be announced in the next IRAF newsletter.

Lindsey Davis

IUE Spectra in IRAF

Recently a site contacted us about getting IUE high dispersion echelle spectra (MEHI in IUE parlance) into IRAF so that such data could be examined and manipulated by the IRAF spectroscopy tasks. In particular they had a FITS format tape consisting of six one dimensional images per echelle order. The main characteristic of the MEHI IUE spectra is that the data arrays are not linearized in wavelength, rather a wavelength coordinate array is supplied. The IRAF spectroscopy tasks either operate upon a dispersion function (generated by IDENTIFY or ECIDENTIFY) or data arrays (images) which have been reinterpolated to linear coordinates (the requirement to reinterpolate the data will be removed someday).

There is no simple mechanism for using a separate wavelength coordinate array. The only way available currently is with ONEDSPEC.SINTERP and that requires doing a lot of manipulations with text files. For the particular IUE echelle data in question I have created a new package, IUEECHELLE, which both provides a conversion to the more convenient IRAF echelle image format and a dispersion linearization task which is nearly identical to the standard task for echelle data (IMRED.ECHELLE.ECDISPCOR) except that instead of a dispersion function a wavelength coordinate image (also in echelle format) is used to define the wavelengths. To complete the package the commonly used tasks for echelle data are also defined in the package. This package is available as an addon external package, but note the following comments.

The FITS data format described above is a preliminary one from the Goddard IUE center. I don't know how long this format will be used or how generally available data will be in this format. The planned final FITS format (currently available from VILSPA) is different and uses FITS tables. I plan to try and follow changes in the data format, although the FITS table extension is not yet supported well in IRAF. Because of these uncertainties about the data formats you should contact me about the applicability of the new package after you have data in hand and before requesting the addon.

Frank Valdes

Image Display on VMS Workstations

Although we do not yet have an image display capability for DECwindows, there is a new version of the UIS display program, which runs under the VAX Workstation Software (VWS). Under VMS 5.1, VWS remains available as an alternative to DECwindows, although switching between the two window systems requires that the VAXstation be rebooted.

Firstly, a **BUG WARNING** ! The version of NEWUISDISP distributed with IRAF 2.8 will not work on 4-bit workstations, due to a feature I added for 8-bit systems and forgot to surround with a check for 8 bits (which is included in more recent versions). To fix it, locate the line in the source code which states **`NNUMBER=10`** and replace it with **`NNUMBER=0`**. You'll have to recompile and relink, as discussed in the comments at the beginning of the source code. Sorry about that.

Version 2, renamed UISDISP in honor of the enhancements (and because everyone was so disparaging about the first name), provides a few minor extras, but most significantly it now has a graphics overlay capability. This means that images can be labelled with text and lines, and that cursor positions can be marked. Full details can be found in the file UISDISP.TXT which, along with the new program, can be found in the directory 5355::USR0:[SHARP.UISEXP]. There are several executables labelled by VWS version (see Bug 1 below), but in case of doubt, recompile and relink on your own system. Interested users without SPAN access should send mail to me (sharp@noao.edu) to find the best way to get hold of the program. Before you leap at this opportunity (!), please read the following.....

<u>Bugs</u> <u>and</u> <u>limitations</u>

1) There is a bug in VWS 3.3. For certain image sizes (such as 128 square), if you enter graphics overlay, place some graphics (text or line), and immediately remove it (by clicking twice on the "place here" button for text, or by clicking the "reject" button for a line), the window would hang. Fixing this requires a reboot. I have installed a workaround, which means your image will reload and flicker more than it should, but at least it doesn't seem to freeze any more. I cannot, however, guarantee that there isn't some other combination of image size and graphics which can trigger this bug. If you upgrade to VMS 5/VWS 4, these workarounds can be removed (look for the code section starting with `BUG AVOIDANCE`). Because of this, I decided to keep more than one executable in the directory mentioned above. These executables are called UISDISPnn.EXE, where nn is the VWS version number (33 or 41, so far).

2) The graphics 'objects' are separate from the image, and do NOT zoom and pan. You should set the image the way you want it before writing the graphics. In fact, any activity which redraws the image will erase all existing graphics: this includes changing the min/max values for the display.

3) The markers are scaled to the image pixels, so that a "single pixel dot" will fill the entire pixel whatever the zoom or replication factors. I consider this a feature rather than a bug, but some may disagree.

4) All coordinates read from files are interpreted as image coordinates. Graphics objects outside the current displayed section of the image will not be displayed, with no warning message, but they are in fact created (which bears on the memory limitation mentioned below). If you want to place graphics objects on the look-up table wedge (?), you have to use the cursor, but you can't put markers there.

5) Memory is even more of a problem than ever before. The graphics items are held as separate objects (enabling selective erase, different colors, line styles, and fonts), each of which requires memory. The "snap" feature has been enhanced to allow the graphics to be included, and this requires that the entire display be mapped, which takes even more memory. You will find that a page file quota of 20000 is a minimum, and is not enough if the program is run as a sub-process, with the quota being shared. The default for snaps is therefore NOT to include graphics.

6) There may be a very small positioning hiccup apparent in the cursor read, region dump and box corner options. With markers enabled, boxes are placed around the region: these boxes are correct, but may look slightly off-center to the eye. The output and the boxes definitely refer to the same area, even if you think it looks funny.

Briefly, the additional features are:

a) Graphics: overlay using specified coordinates or using the cursor, with a choice of color. Text using any of the fonts found in the SYS$FONT directory, with scaling of size or of width and height independently. Lines with a choice of thickness and style, with some predefined styles as well as a "do it yourself" style. Markers for the cursor read mode in one of nine styles, with an option to read marker positions from a file. Markers can be disabled: with them on, region dumps and box corners are also indicated with a thin-line box in the current color. Selective erase of graphics "objects", where an "object" is defined to be an associated set (not just a single marker or a single line). Option to read a completely general file, containing all style declarations and all graphics objects. Option to write out current style settings for use with a general file.

b) an extra look-up table option, being a V-shaped greyscale (i.e. from white to black and back to white, or the reverse).

c) the "snap" option will now create either a single black-and-white image, or three separate images for red, green and blue separately, for use with output devices which think that way.

d) a new "hard copy" option, which creates a UIS-format file which can be converted to Postscript, Regis, Sixel or HPGL using the command `RENDER` (you need to have installed the HCUIS option of VWS).

e) a change to the size limitation: I still don't provide image shrinking or sub-sampling of rasters which are too large, but at least within the current limitations (about 1000x800) the cursors are always correct.

f) some minor bug fixes for odd hiccups I never managed to trigger here, and a revised cursor position algorithm which is more accurate at large zooms.

Finally, for those of you using (NEW)UISDISP on a VAXstation with DECnet to a machine with much more disk space, the original guidelines note that you can read images over DECnet provided that the pixel files and header files are in the same directory. In fact, there's a way to fiddle most cases where the pixel files are in a different directory. After reading the header file, the program complains that it cannot find the pixel file. This is usually because the pixel file name contains a disk specification which is not on the local VAXstation. However, if you define this disk name to be a logical pointing over DECnet, the pixel file can be found. For example, when I specify a file called DRACO::USR0:[SHARP]K123J7A (IRAF file k123.a), I get back an error message referring to file SCR$1:[SHARP]K123J7A.PIX. But if, either before I start UISDISP or from the parent process if I started it as a subprocess, I issue the command DEFINE/JOB SCR$1 DRACO::SCR$1: (where the placement of the colons is very significant), the file will be found and read.

Nigel Sharp

How to Make Portrait Plots

Several users have asked about making rotated plots with a *portrait* aspect ratio, instead of the usual *landscape* plots. This can be easily accomplished using the GKIMOSAIC task in the plot package. This method should produce rotated plots on any given plotter without requiring any reconfiguration of IRAF. The steps are:

1. Save the plot(s) in a metacode file. The graphics may be redirected from the command line (`... >G filename.gki`), may be written from cursor mode (`:.write filename.gki`) or may be generated as the output of certain tasks.
2. If necessary, extract a subset of the plots from the metacode file using GKIEXTRACT and pipe them to the next step.
3. Run GKIMOSAIC on the metacode file or from the pipe mentioned above. The critical parameters are:

   rotate = yes
   nx = **ny** = 1 (to rotate a single plot)
   fill = yes (for multiple plots if **nx** doesn’t equal **ny**)

4. The sense of the rotation is such that the input list may need to be reordered; the plots are rotated 90 degrees clockwise, *individually*. Also, **nx** and **ny** may need to be swapped.

Examples:

1. To make a portrait surface plot of the IRAF test image:

```
pl> surface dev$pix >G m51.gki
pl> gkimosaic m51.gki nx=1 ny=1 rotate+ dev=stdplot
pl> delete m51.gki
pl> gflush
```

2. To make portrait plots of a spectrum using a file of cursor commands:
(The range "2-" includes all of the plots in the file *spec.gki*, except for the first.)

```
pl> splot spec.0001 cursor=curfile >G spec.gki
pl> gkiextract spec.gki "2-" |
>>> gkimosaic spec.gki nx=1 ny=1 rotate+ dev=stdplot
pl> delete spec.gki
pl> gflush
```

the file *curfile* might contain:

```
3000 0 1 a
4000 0 1 a
4000 0 1 a
5000 0 1 a
5000 0 1 a
6000 0 1 a
6000 0 1 a
7000 0 1 a
```

to select four autoscaled one thousand Angstrom regions.

Rob Seaman

Using and Customizing the EPARAM and EHISTORY Editors

Many users seem unaware of the capabilities of the IRAF history editor and of its close kin, the parameter editor. These editors are powerful tools for minimizing the number of keystrokes needed to perform both ordinary and sophisticated functions within IRAF.

The IRAF **history mechanism** has two modes: immediate execution, much like the **!** syntax of the UNIX C-shell; and command line editing, much like the ↑ syntax of the VMS DCL.

The immediate execution mode is simple but powerful:

`cl> ^`	Rerun the last (immediately previous) command.
`cl> ^nite1^nite2^`	Rerun the previous command, replacing the first instance of `nite1` with `nite2`.
`cl> ^nite1^nite2^g`	Do the same, replacing all instances of `nite1`. (`g` for *global*)
`cl> ^rfits`	Rerun the last command that begins with `rfits`.
`cl> ^?FLAT`	Rerun the last command that contains `FLAT` anywhere. (`?` as in a UNIX "backwards search")
`cl> ^?FLAT:p`	Recall the same command to the top of the history buffer and display it on the terminal, but do not execute it. This is handy to prepare the command for editing. (`p` for *print*)
`cl> history`	List the last few commands in the history buffer.
`cl> history 50`	List the last fifty commands in the history buffer.
`cl> ^37`	Rerun number thirty-seven from this list.
`cl> ^-2`	Rerun the command **before** the previous one.
`cl>` *`task`* `^2 ^1`	Run *`task`* with the first and second arguments of the previous command reversed.
`cl>` *`task`* `^$`	Run *`task`* with the last argument of the previous command.
`cl>` *`task`* `^*`	Run *`task`* with all of the arguments of the previous command.

Note that the CL parameter *ehinit* must include `noverify` as one of its options. If `verify` is set, the '`^`' commands will behave identically to the '`e`' commands described below.

IRAF command line editing is achieved with the EHISTORY task. This is in contrast to VMS DCL command editing which is entered by typing an ↑ keystroke (although once you are within EHISTORY, an ↑ will recall successive command lines). To simplify usage, the single character abbreviation, `e`, is reserved for the history editor even though there are other tasks beginning with "e". The various options listed above also work with EHISTORY, for example:

```
    cl> e rfits
```

will recall the last use of the RFITS task and enter the command line editor.

The keystroke directives are identical for EHISTORY and EPARAM. They include the normal things such as the moving, deleting and undeleting of characters, words and lines. The keystroke bindings for the various directives may be configured for your site or for an individual user by changing the appropriate *edcap* file (the originals are in the `dev$` directory).

Here are a few hints and cautions:

- A task may be run directly from within EPARAM. After setting the correct value for each of the parameters (including query parameters) just type `:go` to execute the task. This avoids having to exit the parameter editor (with `^Z`, `^D`, or `:q`) and type the taskname, either specifying the query parameters again on the command line or omitting them and being prompted. Job control and I/O redirection are currently unavailable with `:go`. Other *colon* commands are described in the EPARAM help page.
- In EPARAM, an old parameter entry may be edited without retyping the entire line. The UNDEL_LINE keystroke(s) (`^U^L` in `dev$vi.ed`) will recall the value so that it may be edited, placing the cursor at the right hand side of the line.
- Interactive help is available from within EHISTORY as well as EPARAM. The same keystroke sequence (`ESC-?` in `dev$vi.ed`) works for both tasks. This sequence can be seen at the bottom right of an EPARAM session or can be read from the appropriate edcap.
- The escape sequences for the arrow keys are not read from the IRAF *termcap* database, but directly from the edcap file. This means that the arrow key escape sequences for each of the terminals that is used at your site should be edited into the edcap file. As a general rule there may be more than one character sequence bound to each editor directive.
- The default `vi.ed` file distributed with the system has some broken keystroke sequences containing **tab** (`^I`) and **xoff** (`^S`) characters that are specially treated by the terminal driver and by EHISTORY itself. This problem can be fixed (until the next release) by customizing the edcap file.
- Editing of multiline command blocks (i.e., **terminal scripts**) is not well supported. It can be attempted by first clearing the terminal screen and then by refreshing (`^R` in `dev$vi.ed`) each line of the command block after entering EHISTORY. Use the ↑ and ↓ keys to move up and down within the block.

To make your own custom copy of an *edcap* file:

- Copy `dev$vi.ed` (or `dev$edt.ed`, etc.) to your IRAF home directory:

```
cl> copy dev$vi.ed home$
```

- Make any desired changes in the file. The unusable keystroke directives in `vi.ed` are: DEL_WORD (`^I^W`), DEL_LINE (`^I^D`) and MOVE_START (`^T^S`).
- Either log out of IRAF and back in again, or reset `editor` to another choice, edit something (with *edit*, *eparam* or *ehistory*) and reset `editor` back again:

```
cl> reset editor = edt
cl> e
    reset editor = vi      (i.e., delete edt and type vi)
```

  This will initialize the editor and read the new edcap file. The new keystrokes will now be used whenever you log in.

Rob Seaman

Password Prompts and IRAF Networking

Users should **abort** a task and flush it out of the process cache if they type an incorrect password while attempting network access. Some IRAF tasks will misbehave if a remote device or file access is attempted and an incorrect password is entered, even if the correct password is eventually entered in response to a later prompt.

The potential damage (this is an extreme case) can include mistakenly deleting a file on the local node instead of a file with the same pathname on a remote node. For this to happen, the user need only type the wrong password repeatedly when prompted by the DELETE task (of course there have to be two files with the exact same pathname on the two machines). More dangerously, the mistaken deletion can occur **with no prompting** if there is an incorrect password in a `.irafhosts` file or in the `dev$hostlogin` file. Users or sites which rely on these files should check the passwords carefully!

The underlying cause of this behavior is that the logic of a given task may branch depending on the success or failure of various file accesses. Each time an access is attempted (and the remote kernel server process is not yet active), the networking software tries to connect to the remote machine. Each try will generate a single password prompt. There is no way for the task to know the reason for a failed access; whether it be a non-existent file, an unreachable host or an incorrect password. This ambiguity in interpretation allows a mischievous path through the task's logic. The precise behavior also depends on the host operating system.

Rob Seaman

The Process Cache

The *process cache* is IRAF's way of limiting the overhead of process startup. IRAF groups the compiled code for several tasks into a single executable file; usually each package corresponds to one executable. When a task is run, the command language will check to see if the executable containing the task is already present in the cache, if so that process is reactivated, if not a new process is created which will be kept alive after the task finishes. There is a limit (typically 3 or 4) to the number of processes in the cache; when the limit is reached the process that has been unused the longest will be flushed from the cache. Processes may be locked into the cache to prevent them from being flushed; this should be used with discretion since it defeats the purpose of the cache and may lead to a lockout.

To see what processes are currently in the cache:

```
cl> prcache
```

To lock an executable into the cache, give the name of a task that is in the executable:
(Note that the **system** process is locked in by default.)

```
cl> prcache dir
```

To flush the process cache:

```
cl> flprcache dir              (flushes dir whether it is locked in or not)
cl> flpr                       (flushes all unlocked processes)
cl> szpr=n                     (initializes the process cache)
```

Rob Seaman

Don't Load Packages in Scripts!

Many users have had problems with scripts that load packages containing the tasks used within the script. This practice is ☞ **not recommended** ☜ as is discussed in the newly revised *An Introductory User's Guide to IRAF Scripts*. Unfortunately, an earlier version of the guide suggested that this was acceptable, and indeed, there was no clear policy on the issue at the time that the guide was written.

Although some bugs are revealed, especially when loading packages within **procedure** scripts, the main reasoning behind this restriction is that the suite of *library* packages required for a script forms part of the execution environment of the script. This environment should be defined before a task or script is executed.

The best way to guarantee a well defined execution environment is to declare script tasks within your own package. This package, which is itself defined by a script file (**not** a **procedure** script), should load all the needed packages; it can also be used to declare your IMFORT and SPP tasks. This is described more in the *An Introductory User's Guide to IRAF Scripts*.

Rob Seaman

FOCAS News

There has been quite a bit of activity concerning FOCAS recently. This article summarizes some of this activity. ESO and the Space Telescope European Coordinating Facility (ECF) organized a meeting with representatives of various software systems which do digital photometry. Before the meeting they collected and distributed a number of test images, called the "Standard Test Images", consisting of various stellar and galaxy fields. Several of the images were contributed by FOCAS users. The intent was to have each system process some of the images to present at the meeting and later be made available for comparison. I processed most of the images with FOCAS in April. I fixed several bugs, made some small additions, and basically verified that everything was working properly in the latest version. The results were archived and a paper was written documenting the processing and the archived results. This paper, *Faint Object Classification and Analysis System -- Standard Test Image Results*, was presented and the results contributed at the Two Dimensional Photometry Systems workshop held in Garching, FRG, on April 17th. Papers given at this meeting are to be published in an ESO proceeding. The FOCAS paper and tape archive are available upon request.

Steve Majewski (U. of Chicago, Yerkes Obs.) initiated and organized a first ever FOCAS User's Group meeting held in conjunction with the Astronomical Society of the Pacific's Centennial Meeting in Berkeley on June 21 - 25. The meeting included a status report on FOCAS, some work done by users, and a discussion of future plans and needs. A survey was distributed to aid in making plans and improving communication. One step in organizing a user's group is to prepare a mailing list (largely electronic mailing). I would like to request all FOCAS users who have not received or returned the survey to contact me for a copy or at least send me your email address (or surface mail address if necessary). My email address is `5355::fvaldes` or `fvaldes@noao.edu`.

Since the first announcement of the availability of FOCAS in the IRAF newsletter there have been several important bug fixes. Very recently there have been some changes to take advantage of the IMTOOL enhancements for zoom and color graphics available with IRAF version 2.8 and described elsewhere in this newsletter. These changes are discussed further below. A summary of the bugs is given below with the date it was fixed.

- The starting line offset feature of DETECT does not work. Fixed July 26, 1989.
- Classification of objects near the image edge by RESOLUTION is wrong and causes crashes on Vaxes but not on Suns. Nonborder objects are correctly classified. Fixed July 3, 1989.
- The automatic sigma calculation by DETECT overestimates the sigma. One usually works around this by manually setting the sigma or adjusting the detection thresholds. Fixed April 5, 1989.

- Matching was not always correct due to an initialization problem. Fixed March 31, 1989.
- REVIEW on Sun Workstations with IMTOOL sometimes hangs due to a change in how the terminal window manager works. If you have had problems with REVIEW this is probably it. Fixed March 30, 1989.
- The REVIEW isophotes are sometimes drawn too small by an integer factor. Make sure CDELT image parameter is 1. Fixed March 28, 1989.
- The splitting levels argument in SPLITS was ignored. Fixed March 27, 1989.
- Various image operators, EXPAND, OPIMG, AVFLD, EXFLD, failed to flush out the last part of the image due to a missing close statement needed when using IRAF image format. Fixed March 27, 1989.
- For large images the REVIEW isophotes were only drawn in part of the image. Fixed March 22, 1989.
- A number of small bugs were fixed in December, 1988.
- The K filter option did not work on Suns and Alliants. Fixed November 4, 1988.
- New better and faster version of the RESOLUTION classifier introduced October 4, 1988.

As mentioned earlier, changes were made both to take advantage of the new features added to IMTOOL for the V2.8 IRAF release and to make FOCAS work with the changes. Note the latter point: if you are getting V2.8 IRAF or even just the new IMTOOL, the older FOCAS will become confused if you zoom the display or load it with an image using the IRAF DISPLAY task (the last loaded coordinate system is remembered because the older FOCAS does not reset the coordinates when it loads an image). The changes made and their usefulness are summarized below.

- The IMTOOL zoom may be used anywhere. The older FOCAS will become confused about coordinates.
- Because IMTOOL supports zoom the pixel replication of a factor of 4 used in object display mode in REVIEW was removed.
- The isophotes drawn by REVIEW are now color coded by the standard object classifications. This is pretty and can be useful.

Some other things to be aware of are known problems with image display. FOCAS currently requires a 512x512 frame buffer (i.e. imt512 configuration) regardless of the window size. If the frame buffer configuration is different or changes then image displays by FOCAS will appear grossly wrong; each line beginning in a different column. A subtle confusion reported by a user is that IRAF can automatically change the size of the IMTOOL buffer based on the value of "stdimage". If you are mixing IRAF and FOCAS display commands make sure "stdimage" is set to "imt512" either interactively or, more mistake proof, in your "login.cl".

In earlier versions of IMTOOL a Sun kernel bug (in the FIFO device driver used by IMTOOL) could make the workstation crash occasionally. The current versions of IMTOOL and FOCAS try to avoid the problem by changing the size of FIFO data transfers, and this seems to have eliminated this problem. Sometimes the communications between FOCAS and IMTOOL may get out of synch causing IMTOOL to get into a state where it can no longer execute commands from FOCAS. The buffer size change has improved things but it still happens occasionally. Just start up a new IMTOOL if this happens. The isophote drawing seems to cause this to happen more often. Finally, with the new zoom feature, if the image is zoomed and REVIEW draws isophotes or a cross only a part of each pixel will appear to be drawn giving a Venetian blind pattern. Refreshing the display, e.g., with a zoom or pan, will fill in the rest.

Frank Valdes

Add-on Software Available for IRAF Version 2.8

The following software packages are available as add-ons to IRAF version 2.8.

- Volume rendering software - this software has been discussed in previous issues of the IRAF Newsletters (October 1988 Number 5 and February 1989 Number 6). Contact Steve Rooke (rooke@noao.edu, 5355::rooke) for further information.
- Gould DeAnza IP8400/8500 display software (VMS only) - contact the IRAF HOTLINE.
- Kernel server kits - those files necessary to access tapes drives on a non-IRAF host or those files necessary to utilize a Sun workstation as a smart terminal/display without installing the full IRAF system (for UNIX hosts only). Contact the IRAF HOTLINE for further information.
- UISDISP display software for VMS Workstations - see accompanying article in this newsletter. For further information please contact Nigel Sharp (sharp@noao.edu, 5355::sharp).
- Radial Velocity Analysis package - available for user testing only (see accompanying article in this newsletter). This software has only just reached the user test stage and should not be requested for routine scientific use. On the other hand, if you are concerned about the detailed capabilities of the package this is the ideal time to try the software out and let us know what you would like to change or add. Contact Mike Fitzpatrick for more information (fitz@noao.edu,5355::fitz).
- IUEECHELLE package - a prototype package to support a particular format of IUE Echelle spectra. See the accompanying article in this newsletter. For further information contact Frank Valdes (fvaldes@noao.edu, 5355::fvaldes).

Erratum

There was an error in the article "Calling IMFORT Procedures from C Programs" in the last issue of the IRAF Newsletter (February 1989 Number 6). The example given on the bottom of page 5 read

```
imopen_ (image, &mode, &im, &ier, &len_image)
```

the correct usage is

```
imopen_ (image, &mode, &im, &ier, len_image)
```

since UNIX Fortran (at least for BSD, SunOS, Ultrix, etc.) passes the allocated length of a character variable as a hidden call-by-value argument.

The Editors

IRAF Version 2.8 Revisions Summary

June 30, 1989

1. Introduction

This revisions notice coincides with the release of version 2.8 of IRAF. The V2.8 release is a general release for all supported IRAF hosts.

The following is a brief description of some of the new features available in IRAF Version 2.8. This is not intended to be an exhaustive list, but rather a brief summary of the major changes since the last major release of IRAF Version 2.5 in July 1987 and subsequent intermediate releases primarily to support Sun/IRAF: IRAF Version 2.6 (February 1988), IRAF Version 2.6+ (March 1988), and IRAF Version 2.7 (December 1988).

More detailed revisions notes are available in the system notes files in the `iraf$doc` and `iraf$local` directories, as well as in the online revisions notes for the various packages.

2. IRAF System Revisions

This document highlights the most notable revisions made to the IRAF core system software for Version 2.8. This is only a revisions summary; no attempt is made to provide detailed technical documentation for each revision, nor is there any attempt to exhaustively summarize all revisions. A complete record of all core system revisions will be found in the *System Notes* for V2.8. Additional information on some of the topics covered below will be found in the various *Installation Guides* and *Site Manager's Guides,* and in the *IRAF User and Technical Documentation* manual sets.

2.1. Copyright notice

Subject to AURA and NSF approval, the IRAF software will be copyrighted sometime during 1989. As a first step in this process, a copyright notice has been added to all core system source files. The notice reads as follows: "Copyright(c) 1986 Association of Universities for Research in Astronomy Inc". We will also be adding a file called COPYRIGHT to the distribution stating the terms of the copyright and associated licensing agreement for the software.

The intent of this action is solely to protect the software from unauthorized commercial exploitation, and the copyright grants, or will grant, the right to copy, modify, and redistribute the IRAF software provided the original copyright notice remains intact, the software is made available in source form, and the rights we grant are passed on with the software. We wish to prevent others, especially commercial firms, from copyrighting IRAF software in their own name and possibly taking away the rights we grant with the software. Granting the right to modify and redistribute IRAF software does not mean we want to encourage people to do so, we merely want them to have the legal right to do so if they feel they need to.

2.2. Major system enhancements

The information in this section is provided primarily for the benefit of IRAF site managers and programmers. The reader interested primarily in science applications may wish to skip ahead. Some systems level familiarity with the current IRAF system is assumed.

2.2.1. Layered software enhancements

A given IRAF installation consists of the core IRAF system, and any number of **layered software products** or **external packages**. The goal of the layered software enhancements introduced in V2.8 is to make layered software products self contained and hence independent of the core system and of other layered software. Examples of layered software products are the NOAO packages, LOCAL, STSDAS, PROS, and so on.

The layered software enhancements make it possible to install or deinstall a layered product by modifying only a single file in the core IRAF system. The core system may be updated without affecting layered software, and vice versa. Since layered products are independent and are simple to install, IRAF can easily be configured with only those packages needed at a particular site. Software developers benefit from the layered software enhancements because the facilities provided for development and maintenance of layered software are equivalent to those provided for development of the core IRAF system and the NOAO packages. User sites benefit because it is easy to extend the system with LOCAL packages of their own making.

Each layered product (usually this refers to a tree of packages) is a system in itself, similar in structure to the core IRAF system. Hence, there is a LIB (global system library), one or more BINs (binary file directories), a Help database, a set of global environment definitions, and all the sources and runtime files, all contained within the same directory tree. Layered software products, in their source only form, are portable without change to any computer which runs IRAF.

2.2.1.1. The hlib$extern.pkg file

This is the file which is modified to install or deinstall layered software products. To install a layered product, one creates a directory to hold the software, restores the files to disk, and edits the `extern.pkg` file to tell IRAF the name of the root package of the layered product, and where the root directory is located. If the layered software is distributed in source only form it will also be necessary to recompile the software, but this is a completely automated process.

2.2.1.2. NOAO and LOCAL packages reorganized

As part of the project to better support layered software, the NOAO and LOCAL packages have been reorganized as layered products. These packages are now structurally equivalent to third party (non-NOAO) packages, except that the directory trees are rooted in IRAF. Both packages are now self contained, with their own LIB, BINs, Help database, etc., and with an entry in `extern.pkg`, like other layered products. The NOAO package serves as a working example of how to configure a layered package. The reorganization of these packages should be transparent to anyone merely using the system.

2.2.1.3. The template LOCAL

The LOCAL package included with the distributed system has been stripped of all NOAO site-local tasks and restructured as a layered product, the *template local*. The template local contains only two sample tasks and is not intended as an end-user package, but rather as a template to be copied and modified by sites to construct their own site dependent LOCAL package. The desire to be able to easily develop and maintain locally added packages was one of the major motivations for the layered software enhancements project, and we hope that sites will realize the significance of this new capability and take advantage of it.

2.2.1.4. CL now supports package level BIN directories

Rather than assuming a global BIN directory for all tasks and packages, the CL now permits multiple BIN directories, each BIN directory being associated with the package of definition and all subpackages of that package (unless they have their own BIN). A new BIN directory is declared with the optional argument `bindir=`*path* in the `package` statement, e.g., in a package script task.

2.2.1.5. MKPKG support for package environments

Layered packages now have their own private LIB, including an environment definitions file (`zzsetenv.def`), mkpkg global include file (`mkpkg.inc`), and, optionally, a mkpkg special file list file for each supported host system, listing files requiring special compilation to work around host compiler bugs or whatever. The full mkpkg environment is formed by reading the IRAF core system environment and mkpkg definitions and include files, followed by the package definitions and include files. Reading of the package environment occurs *only* if mkpkg is called with the "**-p**" flag, or if the variable `PKGENV` is defined in the user's environment.

Another way of expressing this is, when using mkpkg within a layered package, one must now specify the name of the layered package in order to pick up the package environment definitions. For example, to update the MTLOCAL package in NOAO, one would type "`mkpkg -p noao update`" in the `mtlocal` directory. If this is not done compilation errors may result, or the executable may not be successfully installed in the package BIN directory.

2.2.2. Multiple architecture support

A single IRAF system (or layered package) can now simultaneously support any number of machine architectures using multiple BIN directories sharing a single machine independent copy of IRAF. Each BIN directory contains all the object modules, object libraries, and executables for a particular architecture. An architecture can represent either a type of hardware, e.g., sparc, mc68020+f68881, mc68020+ffpa, vax, etc., or a software distinction, e.g., systems compiled with different sets of compiler flags, or different versions of a system. Multiple architectures are now supported both for IRAF execution, and for IRAF based software development, e.g., a single version of IRAF can now be used to develop and run IMFORT programs on both Sun-3 and Sun-4 nodes.

The only case where multiple architecture support is used at the present time is in Sun/IRAF, which is often installed on a heterogeneous network of workstations, e.g., Sun-3s with various hardware floating point options, and Sun-4s. A single copy of IRAF will be configured with several BIN directories, one for each supported architecture, and NFS mounted on all the network nodes which will be using IRAF. There is no reason that this feature need be restricted to use with Sun/IRAF, however.

2.2.2.1. IRAFBIN and IRAFARCH

Starting with IRAF V2.8, the old environment variable `IRAFBIN` has been obsoleted and replaced by `IRAFARCH`. On machines which support multiple architectures, the latter defines the architecture to be used for both IRAF execution and software development. If only IRAF execution is needed the variable is optional, with the best architecture being selected automatically when the CL is started. If one will be doing software development (including IMFORT) it is best to define the variable in the host environment before starting IRAF or doing any host level software development. Typical values of `IRAFARCH` for a Sun workstation are "sparc", "i386", "f68881", and "ffpa".

2.2.2.2. System libraries moved to the BIN directory

As part of the revisions required for multiple architecture support for software development, all object libraries have been moved from the global, architecture independent LIB to the architecture dependent BIN, with the LIB entries being replaced by symbolic links (in the case of Sun/IRAF). This should be transparent to both end users and programmers.

2.2.2.3. New bin.generic architecture

On Sun/IRAF systems, which are distributed configured for multiple architecture support, the system architecture is set to `generic` in the distributed system. What this means is that all architecture dependent files (objects and object libraries) have been removed from the system directories and archived in the file `OBJS.arc` in the BIN directory for each architecture. Rebuilding any of the packages in a system would require restoring the binaries for a particular architecture, e.g., typing "`mkpkg sparc`" at the IRAF root would restore the sparc binaries for the core system on a Sun/IRAF installation. Note that this *only* affects software development for the system in question; software development for external packages or private user software is not affected.

2.2.3. Shared library facility

IRAF version 2.8 adds support for a general shared library facility for UNIX based systems. Although currently only used with Sun/IRAF, this facility is potentially useful for other UNIX based IRAF systems as well (VMS/IRAF already has its own shared library facility).

What the shared library facility does is take most of the IRAF system software (currently the contents of the `ex`, `sys`, `vops`, and `os` libraries) and link it together into a special sharable image, the file `S.e` in each core system BIN directory. This file is mapped into the virtual memory of each IRAF process at process startup time. Since the shared image is shared by all IRAF processes, each process uses less physical memory, and the process pagein time is reduced, speeding process execution. Likewise, since the subroutines forming the shared image are no longer linked into each individual process executable, substantial disk space is saved for the BIN directories. Link time is correspondingly reduced, speeding software development.

With the introduction of the shared library facility, the disk space required for Sun/IRAF is substantially reduced. Due to the increased memory sharing and reduced process pagein times performance is substantially improved, especially on systems like the Sun/386i which has a relatively slow SCSI disk and often limited memory. The disk size of small programs is reduced by up to a factor of ten in some cases, e.g., an executable for a small program that was formerly 250 Kb in size might be as small as 25 Kb if the shared library is used and the shared image symbols are omitted at link time.

2.3. User interface changes

2.3.1. Calling IRAF tasks from the host environment

The IRAF main and zmain were modified to make it easier to call IRAF tasks as host level tasks, i.e., without having to set up a command file and run the process with the standard input redirected. In the new scheme, any extra arguments given on the process command line are passed into the IRAF main as a command buffer containing the IRAF command or commands to be run. For example,

```
    cl> x_system.e netstatus
```

would run the command `netstatus` in process `x_system.e`.

```
    cl> x_system.e count "files=*.x"
```

would run the `count` task, counting all ".x" files in the current directory.

```
    cl> x_system.e count "files=*.x 4>_o"
```

would do the same, redirecting the output at the IRAF main level to the file `_o`.

```
    cl> x_system.e 'directory @pars $nargs=0'
```

would run the `directory` task with the given parameter set, with `$nargs` set to 0. If any of the parameters to a task are omitted the task will query the terminal for them in the usual way, so for example

```
      cl> alias count "$iraf/bin/x_system.e count files="
```

would make the IRAF task `count` available in UNIX, allowing the IRAF template specifying the files to be counted to be either given on the UNIX command line, or prompted for if omitted. Given the above alias, one could enter a UNIX command such as

```
      cl> count 'cl$*.h'
```

This feature is available in all UNIX based versions of IRAF V2.8, but did not make it into VMS/IRAF version 2.8.

2.3.2. Image packing density control (impkden)

Some users have complained about images taking up more disk space than they have to, due to the IMIO feature which conditionally blocks image lines to fill an integral number of disk blocks. This can result in more efficient image i/o but can also make a significant difference in the amount of disk space consumed by an image in some cases.

IMIO can actually support both block-aligned and fully packed images. The decision is made at image creation time and is based on the **image packing density** if image lines are block aligned. If the packing density is too low for a block-aligned image, a fully packed image is created to avoid wasting disk space. The default minimum packing density is 0.6, i.e., up to 40% wasted space before IMIO switches to full packing (no wasted space).

For finer control over the packing density, the user can now specify the optional environment variable `impkden`, the numeric value being the mininum packing density. For example,

```
      cl> set impkden = 1.0
```

would completely disable block-alignment of image lines in IMIO.

2.3.3. User libraries (IRAFULIB)

It is now possible for the programmer (SPP or IMFORT) to specify a private directory to be searched at compile or link time when developing IRAF or IMFORT programs. This is done by defining the path to the directory in the user environment as the variable `IRAFULIB`. When locating a particular file, this directory will be searched *before* the IRAF system libraries are searched, hence this feature may be used to substitute custom versions of files in the IRAF system libraries, e.g., for debugging purposes.

2.3.4. New logical printer device LPR

A new logical line printer or plotter device `lpr` is now supported on all UNIX/IRAF systems. This treats the UNIX task *lpr* as a kind of pseudo-device, leaving it up to UNIX to decide what physical device to dispose of the output to. This default is system dependent, but on some systems can be controlled by defining the variable `PRINTER` in the user environment.

2.3.5. Machine independent help database

The IRAF `help` task uses a precompiled binary database to speed help keyword searching. This file is now machine independent, allowing it to be generated on one system and included in software distributions without having to be recompiled. In addition, as part of the layered software support, `help` now allows each external package to have its own private help database. The first time `help` is run, all such databases are read and linked to produce a database containing entries for all help modules in the core system and all installed external packages. The help database file is the file `helpdb.mip` in the LIB directory of the core system and each external package.

2.3.6. Set terminal type will no longer hangup

On systems, e.g., workstations, which provide virtual terminal windows which can change in size, IRAF may query the terminal at run time to determine the screen size. This query is performed, for example, at login time if the terminal type is set to `gterm` or `sun`. Formerly this could cause the login process to hang indefinitely (i.e., until the user typed return or interrupt) if the terminal did not respond to the size query, as would happen when the terminal type was set improperly and the actual terminal ignored the query. Thanks to the addition of non-blocking raw terminal i/o in V2.8 IRAF, the terminal screen size query will now time out with a warning message to reset the terminal type, if the terminal does not respond to the query within several seconds.

2.3.7. Installing a new version of IRAF obsoletes old user parameter files

The problem of old, obsolete user (`uparm`) parameter files being used with a newly installed version of IRAF, which could lead to "parameter not found" error aborts, has been fixed. The CL now checks the date of the file `utime` in HLIB, and refuses to use the user pfile if it is older than either `utime` or the package pfile provided with the new system. The contents of old user pfiles are merged into the new system pfile, as before, preserving learned parameter values even when the user pfile is obsolete.

2.3.8. @file list bug fixed

The problem of the "@file" (at-file-list) syntax not working when the file in question was not in the current directory has been fixed.

2.4. Programming interface changes

2.4.1. IMFORT pixel directory control

IMFORT has been modified to permit specification of the pixel file directory by the calling program. The modifications are completely upwards compatible, i.e., existing programs linked with the new interface will still create pixel files in the same directory as the header file, with "HDR$" in the image header.

The Fortran programmer may set or query the pixel file directory using the following routines:

```
imsdir (dir)            # set pixel directory pathname
imgdir (dir)            # get pixel directory pathname
```

where *dir* is a Fortran character variable. The value should be either "HDR$" (the default) or a concatenatable host directory pathname (i.e., trailing / required for unix). Once set, the pixel directory will be used for all subsequent image create or rename operations by the calling process.

For example,

```
call imsdir ("/tmp3/pixels/")
call imcrea (image1, axlen, naxis, dtype, ier)
call imcrea (image2, axlen, naxis, dtype, ier)
```

If desired the default pixel directory may be specified in the host environment as `imdir` or `IMDIR` before running the program. IMFORT will check the host environment for this environment variable then use "HDR$" as the default if no host definition is found.

Note that although this is similar to setting the value of `imdir` in the IRAF environment, IMFORT programs are not part of the IRAF environment and are not affected by changes to the IRAF `imdir`. Also, since IMFORT is a host level facility and IRAF networking is not supported, the network prefix (e.g., "node!") is omitted from the pixelfile pathname, and since

IMFORT programs are not necessarily used in conjunction with IRAF, the `".."` (hidden file protection) files are not used to protect against image deletion.

2.4.2. Image display interface: IMD

A new interface IMD has been added to provide a rudimentary facility for interactive image display device control. This is an interim prototype interface which will be replaced by the new display interfaces when the latter become available.

The IMD interface operates by mapping an image display device frame buffer onto an IMIO image descriptor. The display frame buffer may then be randomly edited by normal image i/o operations, e.g., to modify subrasters of the displayed image, or overlay the image with color graphics. The image pixel to display frame buffer coordinate transformation is supported, allowing applications to work in image pixel coordinates if desired. This interim interface is what is used by the new display oriented tasks `imexamine`, `imedit`, and `tvmark`.

2.4.3. Image masks: PLIO, PMIO, MIO

The following new VOS interfaces have been added in V2.8 to provide a general boolean or integer image mask facility.

PLIO	pixel list i/o
PMIO	pixel (image) mask i/o
MIO	masked image i/o (image i/o through a mask)

PLIO is a general interface for storing and manipulating multidimensional integer valued rasters containing regions of constant value (i.e., masks). The masks are stored in a highly compressed form, the size of the compressed mask being a function of the information content of the mask. Both pixel array and range list i/o facilities are provided, as well as a set of general boolean raster operators, e.g., to extract or insert subrasters, AND or OR a source with a destination, do the same through a stencil, draw regions of various kinds (point, line, box, circle, polygon), and so on. See the `PLIO.hlp` file in the PLIO source directory for further information.

An interactive debug program (`plio$zzdebug.x`) is provided for experimenting with masks. Note that PLIO is a stand alone interface and is not tied in any way to IMIO, even though the data structure operated upon is similar to an image matrix.

PMIO is very similar to PLIO except that it is used to associate a masks with an IMIO maintained reference image. Currently, the PMIO mask must be the same resolution as the physical reference image. All coordinates input to PMIO are in the *image section coordinates* of the reference image. Hence, given a physical image and associated mask, one can operate upon both through a user specified image section transparently to the applications program. This includes all PLIO style boolean rasterop operations, as well as mask pixel and range list i/o. The PMIO interface is layered upon PLIO and IMIO, and the calling sequences are identical with PLIO except for the package prefix, and the addition of several new PMIO specific routines.

MIO is essentially an extension of image i/o for pixel i/o through a mask. The central routines are the following:

```
                      mio_setrange (mp, vs, ve, ndim)
     n|EOF = mio_[gp]lseg[silrdx] (mp, ptr, mval, v, npix)
```

One defines a rectangular region of the image with mio_setrange, and then sequentially reads or writes line segments until all pixels visible through the mask have been accessed. This type of i/o should be ideal for most image processing applications which need to operate upon only those pixels visible through a region mask (e.g., a surface fitting task), upon all pixels except those on a bad pixel mask (e.g., any analysis program), and so on.

PLIO (or PMIO) masks may be stored in binary files on disk, the files having the extension "`.pl`". The V2.8 version of IMIO has the capability to treat such masks as if they were images, allowing masks to be easily displayed, used in image expressions, converted to image matrices and vice versa, etc. Applications may do either pixel or *range list i/o* to a mask image via IMIO, if MIO is not suitable for some reason.

2.4.4. Photon images: QPOE, QPIO, QPEX

A new set of VOS interfaces supporting photon or **event list data** are now available. The QPOE interface implements the Position Ordered Event list object, which consists of a general header mechanism plus an event list, wherein the events are little data structures, e.g., the attributes required to describe a photon detection (position, energy, time, etc.). QPOE is designed to efficiently access very large event lists, e.g., several hundred thousand or several million events in size. Builtin event attribute filtering and region filtering capabilities are provided for selecting photons from the event list. These filtering capabilities may be combined with the sampling capability to produce filtered, block averaged image matrices from event lists.

The QPOE interfaces are the following:

QPOE	header and file access and management facilities
QPIO	raw and filtered event i/o
QPEX	event attribute filter mechanism
QPF	IMIO/IKI kernel for image interface to QPOE files

QPOE and QPF add a new image type to the system, with `.qp` file extension. Hence, event list data can be used as input to any of the image processing tasks in standard IRAF, in addition to being analyzed by tasks which deal with the individual photon events. A QPOE image is contained in a single file. When a QPOE file is accessed as an image the interface filters and samples the event list in real time, using a user defined filter, block averaging factor, region mask, and so on, producing the image matrix seen by applications at the IMIO level. The QPOE object may be repeatedly examined with different event filters to view the data in different ways.

The QPOE interface, in addition to providing an event list capability for IRAF, serves as a prototype for the "flex-header" portion of the new image structures project. Many of the capabilities to be provided for image storage under the new image structures are already present in QPOE.

Further information is given in the `QPOE.hlp` file in the QPOE source directory.

2.4.5. File manager: FMIO

A new VOS library FMIO has been installed. FMIO is "File Manager I/O", and is used to implement a simple binary file manager which maintains the file data of so-called "lfiles" (lightweight files) inside a single host binary file. The system overhead for accessing lfiles is much less than that of host files, and many lfiles can be used to store a complex data structure without cluttering a host directory or incurring the inefficiency of accessing host files. FMIO is part of the DFIO project and will serve as the lowest level interface within DFIO; it is also used currently in the QPOE interface. Additional information is given in the README file in the source directory for the interface.

2.4.6. IMIO changes

IMIO is the image i/o interface, the standard IRAF VOS interface for managing all varieties of image data.

2.4.6.1. Mask image support

IMIO now supports a new type of image, the **mask image**, stored as a highly compressed binary (PLIO) file with the extension "`.pl`". Image masks are most commonly used to store information describing selected pixels in an associated data image. An image mask is logically a boolean or integer image, up to 28 bits deep, containing information only on selected pixels or regions of pixels. Masks are stored in highly compressed format, e.g., a simple mask may be stored in only a few hundred bytes of space. Mask images are readable, writable, and randomly modifiable, like ordinary raster images. See §2.4.3 for more information.

2.4.6.2. Photon image support

Support has also been added to IMIO for **event list images**, stored as position ordered event list datafiles using the QPOE interfaces. This new image type has the extension "`.qp`". QPOE images are read-only under IMIO. Subject to that restriction, they may be accessed like any other image by any IRAF image analysis program. Accessing an event list image as a raster image necessarily involves a runtime sampling operation, wherein the events in the region of interest are accumulated into an initially zero image matrix; in the process the event list may optionally be filtered by event attribute or event position, e.g.,

```
cl> display "xray.qp[t=(30:40),pha=10,block=4]"
```

would display the QPOE image `xray.qp` with a blocking factor of 4, selecting only those events with `t` (time) in the range 30 to 40 and for which `pha` (energy) has the value 10. The event attributes and their names are user definable and may vary for different types of data. See §2.4.4 for more information.

2.4.6.3. IMPUTH

A new procedure `imputh` has been added to the IMIO header access library. The new procedure is used to append FITS like HISTORY or COMMENT cards to the image header.

2.4.6.4. IMPARSE

The calling sequence of the internal IMIO procedure `imparse` has changed. Although this procedure is internal to the IMIO interface and is not supposed to be used within applications, there may be applications which make use of this procedure. Any such applications must be modified to reflect the new procedure calling sequence or runtime problems are guaranteed.

2.4.7. Null string environment variables

The semantics of the VOS procedures `envgets` and `envfind` have changed. This could affect existing programs and any programs which use these functions should be checked to make certain they will still work properly.

These procedures, used to fetch the string values of environment variables, return the length of the output string as the function value. Formerly, a value of zero would be returned both when the named variable existed but had a null string value, and when the variable was not found. This made it impossible to discriminate between the case of a variable not being defined, and one which is defined but has a null value. The routines have been changed to return the value ERR (a negative integer) if the variable is not defined. Programs which do not wish to make the distinction between undefined and null-valued should check for a function value less than or equal to zero. Programs which check for a function value equal to zero will fail if the named variable is not defined.

2.4.8. Environment substitution in filenames

The VOS filename mapping code has been modified to add support a powerful new environment substitution syntax. Previously the only environment substitution mechanism available was the logical directory facility, which could only be used to parameterize the directory field. The new facility may be used to perform environment substitution anywhere in a filename. This is used in IRAF version 2.8 to implement multiple architecture support, e.g.,

```
cl> set bin = "iraf$bin(arch)/"
```

is how the core system BIN is defined in V2.8 IRAF. The syntax "`(arch)`" tells the filename mapping code to substitute the string value of the environment variable `arch`, if defined. If the variable is not defined the null string is substituted. Hence, if the host system does not implement multiple architecture support and `arch` is not defined, BIN is defined as "`iraf$bin/`", which is the backwards compatible definition. If `arch` is defined as, e.g., "`.vax`", then BIN is defined as "`iraf$bin.vax/`". The new feature allows use of a single environment variable to define the architecture, not only to form filenames, but for other purposes as well, e.g., to generate compiler switches or to control library searching in `mkpkg`.

2.4.9. Nonblocking raw terminal i/o

The VOS file i/o interfaces have been modified to add support for nonblocking terminal i/o. This facility makes it possible to, in effect, "poll" the terminal to see if there is any input waiting to be read, to allow interaction without having a program block if the user has not typed anything.

The immediate application of this in version 2.8 was the modification of the `stty` (set-terminal) facility to implement a time out for the terminal size query. Formerly, `stty` would hang up indefinitely when the terminal type was set to "gterm" but the actual terminal was something different, causing the screen size query to be ignored.

In the more general case, nonblocking terminal i/o makes possible a new class of user interface, which is not only interactive, but **event driven**. Nonblocking i/o makes it possible for an application to be continually processing, while checking the terminal occasionally to see if the user has input any commands.

At present, nonblocking i/o is always used in conjunction with raw mode input from a terminal. A new flag `F_NDELAY`, defined in `<fset.h>`, is used to enable or disable nonblocking i/o. For example,

```
call fseti (fd, F_RAW, YES)
```

enables conventional blocking, single character raw mode reads, and

```
call fseti (fd, F_RAW, YES + F_NDELAY)
```

enables nonblocking raw mode input (`YES`, `NO`, and `F_NDELAY` are bit flags). These modes are mutually exclusive, e.g., the first call may be issued while nonblocking raw mode is in effect to make the reads block, and vice versa. A call to `fset(fd,F_RAW,NO)` disables both raw mode and nonblocking mode. Once nonblocking raw mode is in effect one merely reads characters from the terminal in the usual way, using `getc`. EOF is returned if a read is performed when no data is available for input, otherwise the next character is returned from the input queue. Further information on nonblocking i/o is given in the system notes file.

2.4.10. Function call tables (ZFUNC)

IRAF has always had the ability to compute the integer valued address of a procedure, store that address in a table, and later use the address as an argument to one of the `zcall` kernel primitives to call the addressed procedure. This facility has been extended by the addition of a set of `zfunc` primitives, used to call integer valued *functions*. Only integer valued functions are supported (in order to simplify the kernel support required), but in the systems oriented applications where procedure call tables are used, this is unlikely to be a serious limitation.

2.5. Sun/IRAF specific revisions

2.5.1. IEEE exception handling

By default the IEEE hardware is now configured, on all Sun systems, with the invalid, overflow, and divide by zero IEEE exceptions enabled, and with the default rounding direction and precision modes (nearest, extended) in effect. This configuration should ensure that all questionable floating point operations are detected, and that no IEEE "funny numbers" (NaN, Inf, etc.) get into the data. These values, since they don't behave like ordinary numbers, can cause programs to misbehave, e.g., go into an infinite loop. In Sun/IRAF V2.8, if a computation results in an IEEE funny number being generated, an exception abort will result. The most common example is divide by zero.

The IRAF/IEEE interface offers a special debug feature that may be of interest to programmers developing numerically sensitive software. If desired, one can change the default rounding direction and precision (e.g., to test the numerical stability of applications) by using the debugger to set a nonzero value of the variable `debug_ieee` before running an executable. The procedure for doing this is documented in the system notes file.

2.5.2. IMTOOL enhancements

A number of enhancements and bug fixes have been made for V2.8 to the SunView based IMTOOL image display server. The most notable changes are summarized here; refer to the IMTOOL manual page for a more complete description of the new features.

2.5.2.1. Software ZOOM added

IMTOOL, which has had for some time the ability to pan about on a large image, now has the ability to zoom as well. Both pan and zoom are controlled very conveniently by the middle mouse button: place the mouse on an object and tape the middle button once to pan the object to the center of the display window; tap it again and the image will be zoomed. Zoom, currently implemented by writing directly into the hardware frame buffer, is very fast, almost as fast as a normal unzoomed window refresh. The default set of zoom factors is 1,2,4,8 after which the sequence wraps around to 1. The zoom factors are user configurable via the IMTOOL setup panel; very large zoom factors, e.g., x64, are possible. Dezoom (making a large image smaller) is not currently supported.

2.5.2.2. WCSDIR eliminated

The host level `WCSDIR` environment variable, and the text file used to communicate image coordinate (WCS) information between the display task and the display server, have been eliminated. All WCS information is now passed via the datastream used to pass commands and data between the client and the display server. This eliminates the need for users to have to remember to define `WCSDIR` in order to get coordinates in image units, and some subtle process synchronization problems are eliminated as well.

In a related change, the frame buffer configuration index is no longer passed in during a frame erase, hence it is no longer necessary to erase a frame before displaying an image to ensure that a frame buffer configuration change is passed to the server. The configuration index is now passed when the WCS information for a frame is set.

2.5.2.3. Graphics colors

IMTOOL now allocates a range of pixel values for use as graphics overlay colors. Setting a frame buffer pixel to one of these values causes it to always be displayed with the assigned color. The graphics color values are not affected by windowing the display. The most common use of graphics colors with V2.8 IRAF is for drawing graphics into a displayed frame with the new `tvmark` task, available in PROTO. See the IMTOOL manpage for a table listing the color index assignments.

2.5.2.4. New imtoolrc entries

Several new predefined frame buffer configurations have been added to the default `imtoolrc`. These include an 128 pixel square frame buffer (`imt128`), a 256 pixel square frame buffer (`imt256`), and a full screen display with the same aspect ratio as a 35 mm slide (`imtfs35`).

2.5.2.5. System crash (FIFO) bug fixed

Versions of SunOS through at least 4.0.1 have a bug in the FIFO driver code which can cause the internal kernel FIFO data buffer to be deallocated while it is still in use. This will result in a bad kernel which will eventually panic and reboot the system. This is the cause of the IMTOOL crash problem which some sites may have experienced. IMTOOL has been modified to avoid the circumstances (repeated 4096 byte transfers) which cause the bug to surface. So far as we know, the real bug (in SunOS) has not yet been fixed, but at least on the NOAO systems, the frequency of occurrence of the system crashes is greatly reduced with the new version of IMTOOL which incorporates the workaround for the SunOS bug.

2.5.2.6. Cursor marking now disabled by default

When the interactive image cursor read facility was first added to IMTOOL, the default response to each cursor read was to draw a small white dot at the position of the cursor. This is convenient when marking a series of objects to make a list, but with the increasing number of IRAF programs making user of the interactive image cursor, it has been necessary to change the default to disable automatic marking of each cursor read. The cursor mark feature is still available as an option and can be enabled via the setup panel.

2.5.2.7. Ctrl/b may be used for manual blinking

In addition to the list of blink frames and the timed blink feature IMTOOL has provided for some time, it is now possible to manually cycle through the blink frames with the <ctrl/b> key. Typing <ctrl/b> while the mouse is in the image window will cause the display to display the next blink frame in sequence.

2.5.2.8. F4 key will now toggle setup panel

The F4 function key on the Sun keyboard may now be used to toggle whether or not the setup panel is displayed. This provides a single keystroke alternative to calling up the setup panel with the frame menu.

2.6. VMS/IRAF specific revisions

2.6.1. NEWUISDISP added to VMS/IRAF

Nigel Sharp's `NEWUISDISP` display program, used for image display under UIS on microvaxes with bitmapped displays, is now available in the standard VMS/IRAF release, in the directory `[IRAF.VMS.UIS]`.

2.6.2. New INSTALL.COM script

A new `INSTALL.COM` script (also written by Nigel Sharp) has been added to VMS/IRAF. This script, run by the system manager to install selected IRAF executable images, will now automatically check for and deinstall any old versions of the executables before installing the new ones.

2.6.3. VMS 4.7/5.0

Testing of the standard V2.8 VMS/IRAF release, which was prepared on VMS 4.7, on a VMS 5.0 system has thus far not revealed any problems (NOAO is still running VMS 4.7 as our standard system). Hence it appears that the standard V2.8 VMS/IRAF will *run* under VMS 5. It is likely, however, that any attempt to *recompile* VMS/IRAF under VMS 5 would cause problems, since we have not yet tried to rebuild IRAF under VMS 5, and such a major operating system upgrade will often require changes to the IRAF code. The system may be relinked under VMS 5 if desired, and this does not appear to cause any problems, but neither does there appear to be any benefit to be gained from doing so.

2.7. Summary of IRAF System Packages Revisions

- The tasks RFITS and WFITS in the DATAIO package now support the reading and writing of arbitrary sized data blocks (IRAF version 2.7 and later).
- Several new tasks were added to the IMAGES package. IMCOMBINE (IRAF version 2.6 and later) provides for the combining of images by a number of algorithms. The new task CHPIXTYPE (IRAF version 2.7 and later) changes the pixel types of a list of input images. The task IMSLICE slices images into images of one less dimension (IRAF version 2.8). The task IMSTACK has been moved into the IMAGES package (although it still resides in PROTO as well).

  The IMSTATISTICS task has been rewritten and now allows the user to select which statistical parameters to compute and print (IRAF version 2.8). The IMRENAME task has been modified to allow "in place" image renames, used chiefly for moving the pixel files to a new IMDIR.

  Several other tasks in the IMAGES package were modified (IRAF version 2.8). IMSHIFT was modified to accept a list of shifts from a file. REGISTER and GEOTRAN were modified to accept a list of transforms instead of only a single one. IMHISTOGRAM has undergone extensive revision including support for "box" type plots, support for linear or log scaling in the y coordinate, as well as support for antialiasing of the histogram bins.
- All the tasks in the IMAGES.TV package were modified (IRAF version 2.8) so that if a task is used with an unsupported display device a message is printed to that effect.
- The STTY task in the LANGUAGE package has been improved (IRAF version 2.6 and later) to better facilitate its "playback" feature. These changes have been documented in the online help for the task. This feature is little used by external sites but can be a very useful instructional aid if users are aware of its capability.
- A new task PVECTOR was added to the PLOT package that allows one to plot an arbitrary vector in a two dimensional image (IRAF version 2.6 and later).

  The task STDPLOT was modified (IRAF version 2.8) so that it uses the more popular SGI kernel rather than the NSPP (NCAR) kernel (STDPLOT is now equivalent to the SGIKERN task). A new task NSPPKERN was added that uses the NSPP kernel.
- Two new tasks were added to the SYSTEM package (IRAF version 2.8). The task DEVICES simply prints the `dev$devices.hlp` file as edited by the site manager listing available devices on the local host or network. The REFERENCES task is used to search the help database for all tasks or other help modules pertaining to a given topic, e.g., `references vector` will list all tasks that have the string "vector" in their one line description.

2.8. Glossary of New Tasks in the IRAF System Packages

Task		Package		Description
chpixtype	-	images	-	Change the pixel type of a list of images
devices	-	system	-	Print information on the locally available devices
imcombine	-	images	-	Combine images pixel-by-pixel using various algorithms
imslice	-	images	-	Slice images into images of lower dimension
imstack	-	images	-	Stack images into a single image of higher dimension
nsppkern	-	plot	-	Plot metacode on a NSPP (NCAR) plotter device
pvector	-	plot	-	Plot an arbitrary vector in a 2D image
references	-	system	-	Find all help database references for a given topic

In addition, there are new image display oriented tasks `imexamine`, `imedit`, and `tvmark` in the PROTO package in NOAO (used to interactively examine and edit images, or draw graphics into image display frames). These really belong in the core system but have been placed in `noao.proto` since they are prototype tasks.

3. NOAO Package Revisions

Some of the major revisions to the NOAO packages are listed below.

3.1. Summary of NOAO Packages Revisions

3.1.1. New NOAO Packages

Several new packages have been added to the NOAO suite of packages.

- The APPHOT package is a set of tasks for performing aperture photometry on uncrowded or moderately crowded stellar fields in either interactive or batch mode. This package is now installed in the DIGIPHOT package (IRAF version 2.7 and later). The APPHOT package was available as an add-on package to IRAF version 2.5 and later while it was undergoing alpha testing. Many new features have been added to the package since it first became available including the new task QPHOT (quick aperture photometry) and interaction with the image display cursor for supported image displays (Sun workstation, IIS model 70).
- The CCDRED package provides tools for the easy and efficient reduction of CCD images. This package has been installed in the IMRED package (IRAF version 2.6 and later). The CCDRED package was also available as an add-on to IRAF version 2.5.

  A short demonstration of many of the tasks in the CCDRED package is provided with the DEMO task in the CCDRED.CCDTEST package.
- The IMRED.ECHELLE package has been replaced with a more sophisticated collection of tasks for reducing echelle type data (IRAF version 2.7 and later). The new ECHELLE package recognizes a new image format in which each extracted echelle order becomes a line in a two dimensional image rather than having a separate one dimensional spectrum for each order, although this old output format is still available as an option. Several new tasks exist for computing and applying a wavelength calibration to the data using the echelle relationship between the orders (ECIDENTIFY, ECREIDENTIFY, and ECDISPCOR) as well as for manipulating the new echelle format (ECSELECT, ECCONTINUUM, and ECBPLOT).
- The IRRED package has been added to the IMRED package. The IRRED package collects together in one place those tasks used most frequently by users reducing IR data such as that taken with the IR imager at KPNO. The IRMOSAIC and IRALIGN tasks were available with IRAF version 2.6 and later. IRMOSAIC takes an ordered list of input

images and places them on a grid in an output image. IRALIGN uses this grid and a coordinate list of overlapping objects from the individual subrasters to produce an aligned output image. The tasks IRMATCH1D and IRMATCH2D were available with IRAF version 2.7 and later. These tasks are similar to IRALIGN expect that the intensities of adjacent subrasters can be matched as well. A script called MOSPROC (IRAF version 2.8) has also been added that prepares a list of images for a quick look mosaic.

- The MSRED package has been added to the IMRED package. The MSRED package is a collection of tasks used for reducing multispectral types of data, e.g. fiber arrays, where the individual spectra are for different objects. Like the ECHELLE package, it also has its own multispectral image format (a two dimensional image in which each line is an extracted spectrum). Several new tasks have been added to the package for wavelength calibration of multispectral data.

3.1.2. Modifications to Existing NOAO Packages

- The ASTUTIL package was reorganized (IRAF version 2.6 and later - see IRAF Newsletter No. 3 for details) and several tasks were added and/or modified. A new task ASTTIMES computes and prints astronomical dates and times given a local date and time. A new task RVCORRECT computes and prints radial velocity corrections for an observation. The tasks PRECESS and GALACTIC were modified slightly using different but more accurate algorithms.

  The new task SETAIRMASS (IRAF version 2.8) computes the effective airmass and middle UT of an exposure. This task was also made available in the TWODSPEC and IMRED packages.

- The two tasks in the IMRED.BIAS package, COLBIAS and LINEBIAS, were modified slightly (IRAF version 2.7 and later) so that the fitting parameters for the overscan region can be set by the user as hidden parameters to the tasks.

- The task COSMICRAYS (from the CCDRED package) was made available in the IMRED.GENERIC package (IRAF version 2.6 and later).

- A new task called SYNDICO has been added to the IMRED.VTEL package (IRAF version 2.6 and later). SYNDICO makes glossy prints on the NOAO Dicomed printer of the synoptic, full disk, magnetograms and spectroheliograms taken at the vacuum telescope at Kitt Peak.

- Modifications were made to the IMRED.DTOI package. These changes have been documented in IRAF Newsletter No. 4.

- Three new tasks in the ONEDSPEC package, REFSPECTRA, SEXTRACT, and SPECPLOT, were made available in the IMRED.COUDE, IMRED.IIDS, IMRED.IRS, and IMRED.SPECPHOT packages.

- Many new tasks and features have been added to the ONEDSPEC package.

  The SENSFUNC task was completely rewritten (IRAF version 2.6 and later) to allow determination of extinction, display of flux calibrated spectra, and many new features for displaying and manipulating the data.

  IDENTIFY, REIDENTIFY and DISPCOR were modified (IRAF version 2.6 and later) so that a dispersion solution from IDENTIFY could be shifted without changing the original shape of the coordinate function (see IRAF Newsletter No. 3 for details).

  A new deblending algorithm was added to SPLOT (IRAF version 2.7 and later). See the online help for SPLOT as well as the article in IRAF Newsletter No. 4.

  The tasks in the ONEDSPEC.ONEDUTIL package were absorbed into the ONEDSPEC package (IRAF version 2.7 and later).

The EXTINCT task disappeared with its functionality being taken over by a rewritten CALIBRATE (IRAF version 2.7 and later).

The COEFS task was moved to the IMRED.IIDS and IMRED.IRS packages since this is a very instrument specific task (IRAF version 2.7 and later).

Three new tasks were added to the package. SEXTRACT (IRAF version 2.6 and later) extracts subspectra from one dimensional input spectra. REFSPECTRA (IRAF version 2.7 and later) takes over part of the functionality of the old DISPCOR task and allows the user to define which arc spectra are to be used in the calculation of the dispersion solution of object spectra. SPECPLOT (IRAF version 2.8) is a new plotting task that allows the compression of many spectra to a page (see IRAF Newsletter No. 6).

- Several new tasks have been added to the PROTO package.

  Four tasks were added to IRAF version 2.6 and later. BSCALE is a task that can be used to linearly scale images by the mean, average, or mode of the image. IRMOSAIC and IRALIGN can be used to combine many frames into one large image. These three tasks are also available in the IMRED.IRRED package. MKHISTOGRAM calculates the histogram of the data in a text file.

  Three new tasks were added to IRAF version 2.7 and later. IMSLICE is a task that slices an image into images of lower dimension. IRMATCH1D and IRMATCH2D are two tasks that allow combining of many overlapping images while matching the background intensities in two different ways.

  Three new tasks have been added to IRAF version 2.8 that allow the user to interact with the image display (for supported display devices, ie Sun workstation, IIS model 70). IMEXAMINE allows the user to interactively examine portions of the displayed image. TVMARK allows the user to mark objects on the image display. IMEDIT allows the user to interactively edit an image.

- The APEXTRACT package in the TWODSPEC package has ungone several rounds of modifications, as discussed in the IRAF Newsletters, No. 3 and 4. These changes included improved techniques and additional options for the extraction of data.

  A new task, APSCATTER, has been added to the package (IRAF version 2.8). This task determines and subtracts scattered light from two dimensional aperture or echelle spectra. The task was also made available from within the ECHELLE package. This task was discussed in IRAF Newsletter No. 6.

3.2. Modifications and Additions to Calibration Data

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 current contents of this directory are best summarized by paging through its README file, e.g.,

```
    cl> page noao$lib/onedstds/README
```

Two additional line lists (used by IDENTIFY) have been added to this directory (IRAF version 2.8). These lists, `vacidhenear.dat` and `vacthorium.dat`, are simply the standard `.dat` files in air wavelengths converted to vacuum wavelengths. The equation used for the conversion as well as the appropriate reference in the literature are contained in the README file.

The `thorium.dat` file has been updated to contain thorium lines from 3180 Angstroms to 9540 Angstroms (IRAF version 2.6 and later). Please see the README file for the source.

Two new directories have been added containing flux information for standard stars (IRAF version 2.6 and later): SPECHAYESCAL and SPEC50CAL. Both of these lists are from Massey et al., 1988, Ap. J., Vol. 328, p. 315.

3.3. Glossary of New Tasks in the NOAO Packages

Task		Package		Description	Note
apscatter	-	apextract	-	Fit and subtract scattered light	1
apselect	-	apphot	-	Extract select fields from apphot output files	
asttimes	-	astutil	-	Compute UT, Julian day, epoch, and sidereal time	
badpiximage	-	ccdred	-	Create a bad pixel mask image from a bad pixel file	
bscale	-	proto	-	Brightness scale images: new = (old-bzero) / bscale	3
ccdgeometry	-	ccdred	-	Discussion of CCD coordinate/geometry keywords	
ccdgroups	-	ccdred	-	Group CCD images into image lists	
ccdhedit	-	ccdred	-	CCD image header editor	
ccdlist	-	ccdred	-	List CCD processing information	
ccdproc	-	ccdred	-	Process CCD images	
ccdred	-	ccdred	-	CCD image reduction package	
ccdtypes	-	ccdred	-	Description of the CCD image types	
center	-	apphot	-	Compute accurate centers for a list of objects	3
centerpars	-	apphot	-	Edit the centering parameters	3
combine	-	ccdred	-	Combine CCD images	
cosmicrays	-	ccdred	-	Detect and replace cosmic rays	4
daofind	-	apphot	-	Find stars in an image using the DAO algorithm	
darkcombine	-	ccdred	-	Combine and process dark count images	
datapars	-	apphot	-	Edit the data dependent parameters	3
demo	-	ccdtest	-	Run a demonstration of the CCD reduction package	
ecbplot	-	echelle	-	Batch plots of echelle spectra	
eccontinuum	-	echelle	-	Fit the continuum of echelle spectra	
ecdispcor	-	echelle	-	Dispersion correct spectra	
ecidentify	-	echelle	-	Identify features in spectrum for dispersion solution	
ecreidentify	-	echelle	-	Automatically reidentify features in spectra	
ecselect	-	echelle	-	Select and extract apertures from echelle spectra	
fitpsf	-	apphot	-	Model the stellar psf with an analytic function	
fitsky	-	apphot	-	Compute sky values in a list of annular or circular regions	
fitskypars	-	apphot	-	Edit the sky fitting parameters	
flatcombine	-	ccdred	-	Combine and process flat field images	
flatfields	-	ccdred	-	Discussion of CCD flat field calibrations	
guide	-	ccdred	-	Introductory guide to using the CCDRED package	
imedit	-	proto	-	Examine and edit pixels in images	
imexamine	-	proto	-	Examine images using image display, graphics, and text	
imslice	-	proto	-	Slice images into images of lower dimension	
instruments	-	ccdred	-	Instrument specific data files	
iralign	-	proto	-	Align the mosaiced image produced by irmosaic	3
irmatch1d	-	proto	-	Align and intensity match image produced by irmosaic (1D)	3
irmatch2d	-	proto	-	Align and intensity match image produced by irmosaic (2D)	3
irmosaic	-	proto	-	Mosaic an ordered list of images onto a grid	3
mkfringecor	-	ccdred	-	Make fringe correction images from sky images	
mkhistogram	-	proto	-	List or plot the histogram of a data stream	

mkillumcor	-	ccdred	-	Make flat field illumination correction images	
mkillumflat	-	ccdred	-	Make illumination corrected flat fields	
mkimage	-	ccdtest	-	Make or modify an image with simple values	
mkskycor	-	ccdred	-	Make sky illumination correction images	
mkskyflat	-	ccdred	-	Make sky corrected flat field images	
mosproc	-	irred	-	Prepare images for quick look mosaicing	
msdispcor	-	msred	-	Dispersion correct spectra	
msreidentify	-	msred	-	Reidentify features from one multispec image to another	
msselect	-	msred	-	Select and extract apertures from spectra	
observe	-	ccdtest	-	Create an artificial CCD observation	
phot	-	apphot	-	Measure magnitudes for a list of stars	
photpars	-	apphot	-	Edit the photometry parameters	
polymark	-	apphot	-	Create polygon lists for polyphot	
polypars	-	apphot	-	Edit the polyphot parameters	
polyphot	-	apphot	-	Measure magnitudes inside a list of polygonal regions	
qphot	-	apphot	-	Measure quick magnitudes for a list of stars	
radprof	-	apphot	-	Compute the stellar radial profile of a list of stars	
refspectra	-	onedspec	-	Assign wavelength reference spectra to other spectra	5
rvcorrect	-	astutil	-	Compute radial velocity corrections	
setairmass	-	astutil	-	Compute effective airmass and middle UT for an exposure	6
setinstrument	-	ccdred	-	Set instrument parameters	
sextract	-	onedspec	-	Extract subspectra from dispersion corrected spectra	2
specplot	-	onedspec	-	Stack and plot multiple spectra	5
subsection	-	ccdtest	-	Create an artificial subsection CCD observation	
subsets	-	ccdred	-	Description of CCD subsets	
syndico	-	vtel	-	Make dicomed print of daily grams 18 cm across	
tvmark	-	proto	-	Mark objects on the image display	
wphot	-	apphot	-	Measure magnitudes for a list of stars with weighting	
zerocombine	-	ccdred	-	Combine and process zero level images	

Notes:

(1) Tasks also in echelle and msred packages.

(2) Tasks also in coude, iids, irs, and specphot packages.

(3) Tasks also in irred package.

(4) Tasks also in generic package.

(5) Tasks also in coude, echelle, iids, irs, msred, and specphot packages.

(6) Tasks also in imred and twodspec packages.

Doug Tody
Jeannette Barnes