CCLRC / Rutherford Appleton Laboratory **SG/12.1**
Particle Physics & Astronomy Research Council
Starlink Project
Starlink Guide 12.1

R. Morris, G.J. Privett & A.C. Davenhall
30th December 1998

An Introduction to IRAF

Abstract

This document is an introduction to the IRAF software package and its use on Starlink systems. IRAF is a powerful and comprehensive environment for reducing and analysing astronomical data. It was developed at the National Optical Astronomy Observatories, Tucson and is in widespread use around the world. IRAF has its own data file format, command language, on-line help system and programming language. It is a modular system. The basic core, which is always present, provides general facilities for image processing and data reduction. For more specialised tasks additional packages are loaded to augment the core system. Software for processing most sorts of astronomical data is available for the IRAF environment.

Who Should Read this Document?

This document is aimed at people who are considering whether or not to use IRAF or who are using it for the first time.

Revision History

1. 30th December 1998: Version 1. Original version based on the earlier document SUN/179.3, 13th July 1995 by Rhys Morris and Grant Privett (ACD).

Contents

List of Figures

Part I

Overview

1 Introduction

This document is an introduction to the IRAF (Image Reduction and Analysis Facility) software package and its use on Starlink systems. It is aimed at people who are considering whether or not to use IRAF or who are using it for the first time.

IRAF is a powerful and comprehensive environment for reducing and analysing astronomical data. It was developed at the National Optical Astronomy Observatories (NOAO), Tucson and is now in widespread use around the world. IRAF has its own data file format, command language, on-line help system and programming language. It is a modular system. The basic core, which is always present, provides general facilities for image processing and data reduction. For more specialised tasks additional packages are loaded to augment the core system. Software for processing most sorts of astronomical data is available for the IRAF environment. IRAF is available on all the variants of Unix currently supported by Starlink: Digital Alpha/Digital Unix, SUN/Solaris and PC/Linux.

IRAF is one of several major software environments which are available to Starlink users and upon which entire data-analysis campaigns can be based. Some other examples include: Starlink's own large collection of utilities and application packages, the various incarnations of the Figaro package (SUN/86[14]), AIPS (SUN/207[12]) and MIDAS. Each package has its own particular strengths and special capabilities; IRAF, AIPS and Figaro are probably the most extensive.

IRAF is a sound choice for many Starlink data-analysis users, especially where compatibility with overseas collaborators is a requirement. The choice is harder for those wishing to develop major applications of their own, who may be reluctant to adopt IRAF's non-standard programming language, `SPP`, or who feel that the 'flat' IRAF data format is inadequate for their purposes (see Section 5 for further details). Those who need formal guarantees of future support should also be very wary about committing themselves to any package which is not under UK control. However, IRAF is an important weapon in the armoury of the average Starlink data-analysis user and likely to remain so for some years.

The structure of this document is:

Part I – an overview of IRAF,

Part II – some simple worked examples.

We suggest that you read through the overview to gain an impression of what IRAF is about and then try some of the worked examples. If you have a particular aversion to reading manuals you could proceed straight to the worked examples, though we do not recommend this.

The presence of IRAF is not mandatory at Starlink sites, though in practice it is available at most sites. You should check the arrangements at your site with your local site manager. If IRAF is not available at your site it is possible to obtain a copy via the Internet; the following section gives the details.

2 Obtaining IRAF

You can retrieve a copy of IRAF via the Internet by anonymous ftp. It can be downloaded from either the original NOAO site in Tucson or from a mirror (that is, an identical copy) maintained by Starlink at the Rutherford Appleton Laboratory. The latter is quicker and more convenient for sites in the UK. The Starlink mirror is updated from the NOAO original every night and hence can be no more than twenty-four hours out of date. The access details are as follows.

NOAO, Tucson

URL: `http://iraf.noao.edu/`

Anonymous ftp

site:	`ftp.noao.edu`
directory:	`/iraf`

Retrieve file `README` for further details.

Starlink mirror

URL: `http://www.starlink.rl.ac.uk/iraf/`

Anonymous ftp

site:	`ftp.starlink.rl.ac.uk`
directory:	`/pub/mirrors/iraf`

Retrieve file `README` for further details.

There are additional mirrors in Germany and Japan. Installing IRAF is a non-trivial process and will almost certainly require the assistance of your site manager. Some hints on obtaining and installing IRAF are given in Appendix A. Postscript versions (suitable for printing) of all the IRAF manuals mentioned in this document can also be retrieved from the NOAO IRAF site and the Starlink mirror. In practice you are more likely to use the IRAF sites to obtain documents than a complete installation.

If you are not familiar with the ftp utility then ask a colleague or your site manager for assistance. Alternatively, the books by Krol[10] and Thomas[15] include good descriptions.

3 Documentation

A great deal of documentation is available for IRAF. Most of the documentation comprises detailed manuals for the various individual, specialised packages. However, some more general documentation is available and, in particular, there is an introductory guide for new users: *A Beginner's Guide to Using IRAF*[1]. IRAF is a complex and in some ways non-intuitive system and it is well worth taking the time and trouble to learn the basics of its operation before attempting a complicated data-reduction task. The *Beginner's Guide* is an accessible and thorough document and is a good place to start. Even if you are already familiar with IRAF it is still worthwhile having a look at the *Beginner's Guide* because it may well still contain useful information which is new to you. Another useful document is *A User's Introduction to the IRAF Command Language*[13] which describes the IRAF command language, `cl`.

Copies of these documents can be retrieved from the IRAF anonymous ftp sites (see Section 2 for details). The directory specifications given below are relative to the 'top-level' IRAF directory given in Section 2. The *Beginner's Guide* is available as file:

`pub/beguide.ps.Z`

The *Introduction to the IRAF Command Language* is available as file:

`iraf/docs/cluser.ps.Z`

Both of these files are in compressed postscript format. Remember to set ftp to binary mode prior to retrieving them. Once retrieved, they should be decompressed using Unix command `uncompress` (sic) and can then be printed on a postscript printer. Most IRAF documents (but not the *Beginner's Guide*) are kept in directory `iraf/docs`. A complete, up-to-date list of documents is kept in file:

`iraf/docs/README`

which is a simple ASCII text file.

3.1 On-line documentation

The IRAF Web sites contain a great deal of useful information. See Section 2 for the URLs.

3.2 Frequently Asked Questions

An excellent, up-to-date list of Frequently Asked Questions (FAQs) is maintained for IRAF. It is, perhaps, the single most useful IRAF document and you should familiarise yourself with it; it will probably contain the answers to most of your initial queries about IRAF. An on-line version is available at the following URLs:

NOAO, Tucson:	`http://iraf.noao.edu/iraf/web/faq/FAQ.html`
Starlink mirror:	`http://www.starlink.rl.ac.uk/iraf/web/faq/FAQ.html`

A simple ASCII text version (suitable for printing) can be retrieved by anonymous ftp from file:

`iraf/FAQ`

3.3 Newsletter

A *Newsletter* describing recent enhancements and developments of IRAF is published intermittently. To subscribe to this *Newsletter* send an e-mail message to username `iraf@noao.edu` asking to be added to the distribution list and including your e-mail and postal address. Alternatively use ftp to retrieve the registration file:

`iraf/REGISTER`

and return it to username `iraf-requests@noao.edu`.

Back issues of the *Newsletter* are available on the IRAF Web pages. A table-of-contents for all the issues can be retrieved by anonymous ftp from file:

`iraf/docs/TOC_news.txt`

and is available from the Web pages.

4 IRAF Version Numbers

Currently the most recent major release of IRAF (at least for the three variants of Unix supported by Starlink) is version 2.11. However, the previous version, 2.10, is still in use at some sites. There are significant differences between versions 2.10 and 2.11, particularly in the data formats which they support (see Section 5). These differences will be alluded to at various points in the present document.

5 Data File Formats

The basic IRAF data file format is the so-called Old IRAF Format (OIF). Despite this name there is no 'new' IRAF format. Most IRAF tasks use the OIF format. Some brief details of it are given below.

In IRAF version 2.11 and subsequent versions most tasks can also access data files in the widely-used international standard FITS[1] (Flexible Image Transport System) format.

Other formats are used for more specialised purposes. For example, the STF format is used by the STSDAS package for storing HST data, the QPOE format is used to hold photon event lists (which are usually produced by X-ray astronomy satellites) and the PLIO format for pixel lists (which flag individual pixels in an image).

The IRAF `dataio` package contains tasks for converting between numerous different formats.

5.1 Old IRAF Format

A dataset, typically an image or spectrum, in the Old IRAF Format (OIF) format consists of two files: a header file which has file-type `.imh` and a pixel file which has file-type `.pix`. These files are usually stored in standard locations in your IRAF directories. The `.pix` files are large and can be stored in either the same directory as the header files or in a different directory. The header file contains the dimensions of the data array, the name of the pixel file and optionally any auxiliary information describing the image, expressed as a series of keywords and their values. These OIF keywords deliberately have a syntax similar to FITS keywords. The image dimensions and name of the pixel file are stored in a binary format, but the keywords are stored as simple ASCII characters. Thus, you can check the values of the keywords by listing a header file on the terminal.

There are several points about the OIF format which you should be aware (and beware) of.

1. IRAF version 2.10 supports only the original OIF (that is, version 1 of the format). However, version 2.11 also supports an enhanced version of the format (version 2).

[1] The original FITS format was proposed by Wells *et al.*[18]. However, it has been developed and enhanced over the years. The FITS standard is now maintained and documented by the FITS Support Office of the Astrophysics Data Facility at the NASA Goddard Space Flight Center (see URL: `http://www.gsfc.nasa.gov/astro/fits/fits_home.html`). Though FITS is basically an astronomical format it is sometimes mentioned in books about standard image formats. See, for example, *Graphics File Formats* by Kay and Levine[9].

2. In version 1 of the format the `.pix` file differs on different types of computer. That is, if you are using IRAF version 2.10 then a data file written on a SUN workstation cannot be read on a Digital Alpha or vice versa. Thus, in this case, it is sensible to carry out all your IRAF work on the same sort of machine.

3. Version 2 of the format is independent of the host operating system. That is, if you are using IRAF version 2.11 then a data file written on one type of computer can be read on another type.

4. This change is backwards but not forwards compatible. IRAF version 2.11 will read files in either version of the format, but version 2.10 only reads files in version 1 of the format. If you need to write a data file with version 2.11 and subsequently read it with version 2.10 then one approach is to write it as a FITS file. Alternatively, version 2.11 can be made to write files in version 1 of the format by setting an appropriate environment variable:

   ```
   set   oifversion=1
   ```

5. In either version of the format, the header file contains the full name and directory specification of the pixel file. Consequently, IRAF data files cannot be manipulated (copied, renamed or deleted) using standard Unix operating system commands; there are special commands within IRAF (`imcopy`, `imrename` and `imdelete` respectively) which must be used for these tasks. This point is covered fully in the *Beginner's Guide*[1].

5.2 Space Telescope Format

The Space Telescope Format (STF) is used for storing Hubble Space Telescope (HST) data. It was introduced for the STSDAS package (see Section 7.1), but is now understood by most IRAF tasks. Like the OIF format, an STF dataset comprises two files: one is the header file, and has the extension .*xx*`h`, where *x* can be any alphanumeric character, while the other contains the actual data values and has a .*xx*`d` suffix. The most usual file extensions are `.hhh` for header files and `.hhd` for data files. These files are usually kept together in the same directory. STSDAS files have the advantage that the header file is an ASCII file and can be read on a terminal without using IRAF at all (which is not true of `.imh` files; see above).

6 The IRAF Package Structure

IRAF's 'tasks', as the individual programs are usually called, are divided into packages. Each of the packages may contain tasks or further subpackages. For example, loading the `noao` package by typing '`noao`', leads to further subpackages, while loading the `dataio` package makes available tasks for importing and exporting data to and from IRAF. You cannot use any task until you have loaded the package which contains it. Typing '`bye`' unloads the most recently loaded package.

7 Some IRAF Packages

A couple of the more common IRAF packages are described briefly below.

7.1 STSDAS

STSDAS (Space Telescope Science Data Analysis System) is a package for analysing data acquired with the Hubble Space Telescope (HST). It was developed at the Space Telescope Science Institute (STScI) in Baltimore. Further information is available at URL:

`http://ra.stsci.edu/STSDAS.html`

including details of retrieving the software and documentation. Note that installing STSDAS requires about 85 Mbytes of free disk space. Also note that STSDAS requires the TABLES package for manipulating tables to be installed. TABLES is also available from the STScI. Inquiries, queries, bug reports etc. about STSDAS can be sent by e-mail to the HST Help Desk at username `help@stsci.edu`.

7.2 PROS

PROS (Post-Reduction Off-line Software) is a package for analysing X-ray astronomy data. It is not specific to any one X-ray astronomy satellite and, in particular, it has facilities to handle data from the EINSTEIN, ROSAT, ASCA and AXAF satellites. It can also be used for non-X-ray data and is particularly suitable for generating pixel masks. It was developed at the Smithsonian Astrophysical Observatory, Cambridge, Massachusetts. Further information is available at URL:

`http://hea-www.harvard.edu/PROS/pros.html`

including details of retrieving the software and documentation. Inquiries, queries, bug reports etc. about PROS can be sent by e-mail to the PROS user support hotline at username `hotseat@cfa.harvard.edu`.

8 IRAF and Starlink

Most Starlink software is written using the Starlink software environment and stores data in the N-dimensional Data Format (NDF; SUN/33[16]). A substantial degree of inter-operability is possible between this software and IRAF.

At the most basic level Starlink provides the CONVERT package (SUN/55[7]) which is a set of applications for converting data files between a number of different data formats. It includes applications for converting an NDF format file to one in the IRAF OIF format (see Section 5) and vice versa. You should be aware, however, that the NDF format is more capable, and can contain more auxiliary information, than the IRAF OIF format. When an NDF file is converted to the OIF format such information is lost. In particular, the NDF format has standard ways of representing arrays of errors associated with arrays of data and many applications in the Figaro package process and propagate these errors. Error arrays are not accommodated in the OIF format and are lost when an NDF is converted to this format.

In addition to explicitly converting NDF files, most Starlink applications which read and write NDF files can be configured to read and write IRAF OIF files instead. Again see SUN/55[7] for details.

Some Starlink packages have been enhanced so that they can be run from the IRAF command language, `cl`. When used in this way the Starlink packages behave like 'native' IRAF packages loaded on top of the basic IRAF; they read and write data files in the native IRAF format and their parameters can be set and inspected using the usual mechanisms of the IRAF command language. Tasks in the Starlink packages can be intermixed freely with the standard IRAF tasks (again, just like 'native' IRAF packages). There are a few differences between Starlink and 'native' IRAF tasks, principally in the way that graphics are displayed; see SUN/217[3] for details.

The principal Starlink packages which have been enhanced in this way are Figaro, KAPPA and GAIA. Their use from the IRAF command language is described in SUN/217[3] and SUN/221[4] and Section 13 gives a worked example. The details of the technical legerdemain which allows applications written for the Starlink software environment to be run from the IRAF `cl` will not be of interest to most users. However, for completeness it is documented in SSN/35[2]. The Figaro, KAPPA and GAIA packages themselves are documented in SUN/86[14], SUN/95[5] and SUN/214[8] respectively.

9 Obtaining Assistance

For queries about how IRAF is installed at your site or assistance with obtaining and installing it you should consult your local site manager.

If you have a problem with or query about using IRAF then your first port of call should be the FAQ (see Section 3.2), which will often have the answer.

The IRAF Project in Tucson provides a support service for users. If your query is not covered by the FAQ then you can send it in an e-mail message to username `iraf@noao.edu`. The response is usually pretty swift, but do bear in mind the time difference between the UK and Tucson.

Because the support provided by the IRAF Project is excellent, little additional support by Starlink is either called for or provided. However, there is a Starlink contact for assistance with IRAF problems; you should send an e-mail message to username `iraf@star.rl.ac.uk`. This contact is particularly appropriate if you encounter problems inter-operating Starlink and IRAF software.

Part II

Worked Examples

10 Introduction

This part of the document provides a few simple, introductory examples of using IRAF. The examples are:

- starting IRAF,
- a simple IRAF session,
- using Starlink packages from IRAF,
- using scripts and batch jobs.

The best way to use the examples is to work through them in the order that they are given. All the data files, scripts etc. referred to in the examples are available in directory:

```
/star/examples/sg12
```

10.1 Additional examples

The examples given here form only the briefest of introductions and there are many facilities which are not described. There are further examples, and more detailed descriptions, in the *Beginner's Guide*[1].

There are also a set of exercises which demonstrate how to reduce various sorts of data. They can retrieved by anonymous ftp (see Section 2 for details) from file:

```
iraf/misc/exer2102.tar.Z
```

The file is a compressed tar archive. Remember to set ftp to binary mode prior to retrieving it. The file should be decompressed using Unix command `uncompress` (sic). Instructions for installing the examples are given in file `exer2102.readme` (which is included in the tar archive).

11 Starting IRAF

Starting IRAF for the first time is non-trivial and hence this example is provided as an illustration of the procedure. *The example is broadly correct, and works in Edinburgh. However, some of the details, in particular the file locations, may be different at your site. You should check the details at your site with your site manager.*

An additional complication is that IRAF has various ways of displaying two-dimensional images (such as the CCD frames), none of which can be considered standard. Some of the common alternatives are to use either SAOTNG, its predecessor SAOIMAGE (see SUN/166[11]) or X-IMTOOL. SAOTNG and SAOIMAGE are separate image display applications which are not part of IRAF (though, obviously, they can display IRAF images). SAOIMAGE works correctly with IRAF version 2.10, but there seem to be problems with using it with version 2.11. Hence SAOTNG is used in the current example. It is also worth noting that when using IRAF you will not always need to display two-dimensional images, though in practice you usually will. Consequently setting up SAOTNG is integrated with the instructions here. The IRAF *Beginner's Guide*[1] covers starting IRAF for the first time, but not all the details of setting up an image display.

The procedure is as follows.

1. In order to use IRAF you need a 'base' or 'root' directory which will contain various configuration files, sub-directories and probably your IRAF data files. This directory is your 'IRAF home directory' and you need to decide where it is going to be located and what it is going to be called. For example, we might decide that our IRAF directory is going to be:

   ```
   /home/acd/iraf
   ```

   This directory will be used in this example, though if you are working through it you should substitute your own chosen IRAF directory.

2. You will probably need to add a one or more lines to your shell login script. If you run the 'tc' shell then this script is file `.tcshrc`; if you run the 'c' shell it is file `.cshrc`. In either case it is located in your home directory. The details will depend on how IRAF has been set up at your site. At Edinburgh the following line should be added:

   ```
   /net/reaxp02/packages2/iraf211/jbin/login
   ```

   See your site manager for the details at your site. It is probably most convenient to put these lines at the end of the file.

3. Either log out and log in again or `source` your `.tcshrc` (or `.cshrc`) file.

4. If you have not already done so, create your own IRAF directory and move to it (for example, `mkdir iraf` followed by `cd iraf`).

5. Type:

   ```
   mkiraf
   ```

You will be prompted for the terminal type; reply `xgterm`. Subdirectory `uparm` will be created in your IRAF directory, as will file `login.cl`. The latter is the IRAF start up script. It contains various options and defaults which you can change, but you should not need to do so.

6. Customisations can be made in file `login.cl`, but are better put in a separate file, which must be called `loginuser.cl`. (It is better to put customisations in `loginuser.cl` than in `login.cl` because the latter is overwritten when `mkiraf` is executed whereas the former is not.) An example `loginuser.cl` file is available as file:

   ```
   /star/examples/sg12/loginuser.cl
   ```

   You can simply copy this file to your IRAF directory. (It configures IRAF to handle the large headers often associated with fibre spectroscopy data and specifies that the pixel files associated with bulk datasets, such as images or spectra, are to be kept in subdirectory `pixels` of the IRAF home directory. Section 11.2 gives some notes on how to specify the directory in which the pixel files are to be kept.)

7. Start SAOTNG ready for use with IRAF. Type:

   ```
   saotng &
   ```

   The '`&`' is, of course, simply to run SAOTNG as a detached process, which is more convenient.

8. Create an `xgterm` window:

   ```
   xgterm &
   ```

   A new window with a distinctive appearance is created. You will use this window to run IRAF. You can run IRAF from your normal login window, but using an `xgterm` window allows you to make better use of the workstation or X-terminal graphics.

   Depending on the configuration options in place at your site the `xgterm` window might be created without a scroll bar. If so, and you want to add a scroll bar (which is often useful), then from the `xgterm` window hold down the `Ctrl` key and simultaneously click on the right mouse button. A window containing a list of options should appear; simply click on the `scrollbar` item and a scroll bar should be added to the window.

9. Move to the `xgterm` window and type:

   ```
   cl
   ```

   An introductory message should be displayed and the prompt should change to '`cl>`'. You are now running the IRAF command language and ready to issue IRAF commands. See the *Beginner's Guide*[1] for an introduction to the commands available and Section 12 for a simple example.

   As a quick check that the image display is working correctly type:

   ```
   display   dev$pix   1
   ```

   An IRAF test image of the spiral galaxy M51 should appear in the SAOTNG display.

   Type `logout` to leave the IRAF command language and return to the Unix shell prompt.

11.1 Subsequent IRAF sessions

Once you have set up for using IRAF it is not necessary to repeat the entire foregoing procedure every time you use IRAF. For subsequent IRAF sessions in which you will use an image display you need repeat only steps 7, 8 and 9. For sessions in which an image display is not needed just repeat steps 8 and 9.

11.2 Specifying the directory to contain the pixel files

The directory which is to contain the pixel files associated with bulk datasets (such as images or spectra) is specified by the value of environment variable `imdir`. For example, the sample `loginuser.cl` customisation file contains the line:

```
set   imdir=HDR$pixels/
```

which specifies that the pixel files are to be kept in subdirectory `pixels` of your IRAF home directory. Note that the trailing '`/`' is important. Here `HDR$` denotes your IRAF home directory, which contains the header files. If you wanted to keep the pixel files with the header files in your IRAF home directory you would simply specify:

```
set   imdir=HDR$
```

12 A Simple IRAF Session

This example demonstrates a simple session using IRAF. It covers: converting a FITS image to the IRAF OIF format, displaying an image and calculating statistics for an image. It also introduces the IRAF on-line help system and displaying and setting IRAF task parameters.

The FITS image used in this example is available as file:

```
/star/examples/sg12/ccdimage.fit
```

If you intend to work through the example using this image you should make a copy of it in your IRAF directory. Alternatively, you may prefer to use a FITS image of your own.

1. Start IRAF and SAOTNG as described in the previous example (Section 11). Subsequent commands should be issued from the window which is running the IRAF command language (that is, the one with the prompt set to '`cl>`').

2. The first step is to convert the FITS image to the IRAF OIF format (see Section 5). If you are using IRAF version 2.11 this step is not strictly necessary as this version of IRAF can access FITS files directly. However, it is included here in order to keep the example consistent for versions 2.10 and 2.11. In addition, you might find the OIF format more convenient for subsequent operations. You should also note that the command to perform the conversion is slightly different in the two versions. FITS file `ccdimage.fit` (or an alternative) should be available in your IRAF directory. Type either:

   IRAF version 2.10:

   ```
   rfits   ccdimage.fit   1   ccdframe
   ```

   IRAF version 2.11:

   ```
   rfits   ccdimage.fit   0   ccdframe
   ```

   A line similar to the following should be displayed:

   ```
   File:  ccdframe KAPPA - Cadd Size = 384 x 256
   ```

   File `ccdframe.imh` should be created in your IRAF directory and file `ccdframe.pix` in the pixels directory (probably subdirectory `pixels` if you have used the `loginuser.cl` file supplied with this document). From the IRAF command language you can show the files in the current directory with command `dir`, change to a new directory with `cd` and list a text file with `cat`; these commands behave similarly to the equivalent Unix commands. You can use these commands to check that the header and pixel files have been created. When you are satisfied that all is well, make your IRAF directory your current directory again.

3. To show a summary of the file, including its provenance, type:

   ```
   imhead   ccdframe
   ```

4. To display an image using SAOTNG simply type:

```
display   ccdframe   1
```

The image should be displayed in the SAOTNG window. You can use the various buttons in the SAOTNG window to customise the display.

5. To display simple statistics about the image type:

```
imstat   ccdframe
```

Output similar to the following should be listed:

```
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
             ccdframe     98304      508.     308.3      330.    31988.
```

6. On-line help information is available for all the IRAF tasks. To obtain help on `imstat` simply type:

```
help   imstat
```

A description of the command and its various parameters will be displayed. Some brief details of the IRAF on-line help are given in Section 12.1, below. So far, we have been specifying information to IRAF tasks by giving mandatory parameters on the command line. IRAF tasks also have so-called 'hidden' parameters which adopt a default value if you do not specify them. You can show the current value of all the parameters of a task using the command `lpar`. For example:

```
lpar   imstat
```

The output should be something like:

```
   images = "ccdframe"      Images
  (fields = "image,npix,mean,stddev,min,max") Fields to be printed
   (lower = INDEF)          Lower cutoff for pixel values
   (upper = INDEF)          Upper cutoff for pixel values
(binwidth = 0.1)            Bin width of histogram in sigma
  (format = yes)            Format output and print column labels?
    (mode = "ql")
```

The parameters enclosed in brackets are hidden (in this case this is all of them except `images`).

7. By default `imstat` displayed the name of the image, the number of pixels, the mean, standard deviation and minimum and maximum values. However, the help information revealed that other options were available and that these were controlled by the value of parameter `fields`. Suppose that we wished to display the mean, standard deviation, skewness and kurtosis[2]. It is necessary to set parameter `fields` appropriately, which is achieved by typing:

[2] The skewness and kurtosis are standard statistical quantities. See, for example, Wall[17].

```
imstat.fields   =   "mean,stddev,skew,kurtosis"
```

If `imstat` is run again:

```
imstat   ccdframe
```

the output should now be:

```
#      MEAN    STDDEV      SKEW  KURTOSIS
       508.     308.3     62.03     4728.
```

In addition to setting parameter values from the command line they can also be edited using task `epar`; see the *Beginner's Guide*[1] for further details.

8. Finally, type:

   ```
   logout
   ```

   To leave IRAF.

12.1 On-line help

On-line help on individual tasks is available from the IRAF command language.

1. To obtain help on an individual task on simply type:

   `help` *task-name*

   from the IRAF command line. For example:

   ```
   help   imstat
   ```

   and the help text for task `imstat` is displayed. The bottom line of the window gives a summary of the commands available (type `q` to quit and return to the IRAF `cl>` prompt).

2. If you are looking for tasks to perform some function but are not sure what the task names are, which packages they are in, or indeed whether they exist, try typing:

   `apropos` *topic*

   For example, if you were looking for commands to mosaic several CCD frames together you might type:

   ```
   apropos   mosaic
   ```

   This command searches all the available IRAF help databases and lists a summary of all the tasks with (in this example) the word 'mosaic' in their title. Note, however, that `apropos` is part of STSDAS and hence is only available if STSDAS is installed.

13 Using Starlink Packages From IRAF

This example shows a simple session using a Starlink package from within IRAF. It covers loading the package and calculating statistics for an image.

Several Starlink packages are available from within IRAF, of which the principal ones are KAPPA (see SUN/95[5]) and Figaro (see SUN/86[14]). When used from within the IRAF command language these packages behave in most respects like 'native' IRAF packages: they can read and write IRAF OIF format data files and their parameters can be displayed and controlled using the usual mechanisms of the IRAF command language. Tasks in the Starlink packages can be intermixed freely with the standard IRAF tasks (again, just like 'native' IRAF packages). There are a few differences between Starlink and 'native' IRAF tasks, principally in the way that graphics are displayed; see SUN/217[3] for details.

This example uses the Figaro application `istat` to display the statistics of an image in order to compare and contrast it with the standard IRAF application `imstat` used in the previous example. The IRAF OIF image `ccdframe` which was used in the previous example will also be used again here.

1. If you are not already running IRAF then start it (see Section 11). Subsequent commands should be issued from the window which is running the IRAF command language (that is, the one with the prompt set to '`cl>`').
2. The first step is to load the Figaro package. Simply type:

   ```
   figaro
   ```

   A list similar to the following should be displayed:

   ```
   abconv    cset      fscrunch  ilog      lxset     sfit
   abline    cspike    fwconv    image     lyset     slice
   adjoin    delobj    gauss     imult     mask      soft
                                 ⋮
   ```

   Also the prompt should be reset to '`fi>`'. The list comprises all the commands in the Figaro package. If instead of the list of commands you get an error message like:

   ```
   ERROR: task 'figaro' not found
   ```

   then Figaro has not been installed in the version of IRAF available at your site. Section 13.1 below gives a few hints on how to proceed in this case.

   However, assuming that the package loaded successfully, then all the Figaro tasks are now available, in addition to the basic IRAF tasks. Here Figaro is behaving just like a 'native' IRAF package. For any IRAF package, native or Starlink, you can:

   - load it by typing the package name,
   - remove it by typing `bye`,
   - obtain help on it by typing `help` *package name*.

Any number of packages can be loaded simultaneously. All the commands in all the loaded packages are available. Packages are hierarchical, with packages having sub-packages inside 'em, and so *ad infinitum* (as the poet almost had it). Typing `bye` removes the most recently loaded package or sub-package. For example, type:

```
bye
```

Figaro will be removed and the prompt will be reset to `cl>`.

3. Now reload the Figaro package by typing `figaro`. Once the package is loaded simple statistics of an image can be displayed by typing:

   ```
   istat   ccdframe
   ```

   Output similar to the following should be listed:

   ```
   Y-range 1 to 256
   X-range 1 to 384
   Total (over 98304.0 pixels) = 4.9943E+7
   Max   = 31988    in pixel (332,108)
   Min   = 330      in pixel (1,256)
   Mean  = 508.05
   Sigma = 308.263
   ```

   Compare this output with that produced by `imstat` in the previous example. As in this case, Starlink applications are usually somewhat more verbose than standard IRAF tasks.

4. On-line help information is available for the Figaro tasks, as it is for the standard ones. To obtain help on `istat` simply type:

   ```
   help   istat
   ```

   A description of the command and its various parameters will be displayed. Again as for the standard tasks, the parameters of Figaro applications can be shown in the usual way:

   ```
   lpar   istat
   ```

   The output should be something like:

```
          image = ""              (IMage) Name of image to examine
         ystart =                 *! (YStart) First Y value to be used
           yend =                 *! (YEnd) Last Y value to be used
         xstart =                 *! (XStart) First X value to be used
           xend =                 *! (XEnd) Last X value to be used
        (pass2 = no)              (PAss2) Pass twice through data for better sigm
       (median = no)              (MEDian) Calculate median as well as other stat
   (stat_total = 0.)
     (stat_max = 0.)
     (stat_min = 0.)
    (stat_mean = 0.)
   (stat_sigma = 0.)
    (stat_size = 0.)
    (stat_xmax = 0.)
    (stat_xmin = 0.)
    (stat_ymax = 0.)
    (stat_ymin = 0.)
  (stat_xstart = 0.)
    (stat_xend = 0.)
  (stat_ystart = 0.)
    (stat_yend = 0.)
  (stat_median = 0.)
         (mode = "ql")
```

5. In addition to the quantities calculated automatically `istat` may also optionally calculate the median value of the image. Type:

   ```
   istat.median   =   yes
   ```

   Now run `istat` again:

   ```
   istat   ccdframe
   ```

   The output should now be:

   ```
   Y-range 1 to 256
   X-range 1 to 384
   Total (over 98304.0 pixels) = 4.9943E+7
   Max   = 31988    in pixel (332,108)
   Min   = 330      in pixel (1,256)
   Mean  = 508.05
   Sigma = 308.263
   Median  494
   ```

   Note that the median is now calculated and displayed.

13.1 Installing Starlink software in IRAF

IRAF installations at Starlink sites will usually have the additional Starlink packages installed. However, if they appear not to be available at your site then this section gives a few hints on how to proceed.

The directory `/star/iraf` should contain a subdirectory for every Starlink IRAF package present at your site. Check that this directory exists and that there is a subdirectory for the package that you are interested in. If the required subdirectory is absent then ask your site manager to install the package.

If the directory is present but the package is still not available when you start IRAF the most likely explanation is that the package is not known to IRAF as an external package. An entry for the package should be added to the `hlib$extern.pkg` file. Your site manager will have to add this entry. SSN/35[2], Section 4.8 contains the necessary details.

As an interim measure, perhaps to establish the cause of the problem, you can add an entry to your own `login.cl` file. For example, for Figaro add the following lines:

```
reset   figaro            = starlink$figaro/
task    figaro.pkg        = figaro$figaro.cl
```

These lines must be added just before the `user` package definition. The `helpdb` definition can be extended by a command like:

```
reset helpdb = (envget("helpdb") // ",figaro$helpdb.mip")
```

For other packages you would add equivalent lines, substituting the name of the package for `figaro`.

14 IRAF Scripts and Background Jobs

A series of IRAF commands can be executed as a script by putting them in a file and then running the file. The IRAF command language, `cl`, is also a programming language and contains many features which allow you to write scripts to perform repetitive tasks automatically. It is also possible to run IRAF scripts as Unix background jobs. This facility is particularly useful if you want to run lengthy tasks at low priority without tying up your terminal. This example is a simple introduction to IRAF scripts and background jobs.

The scripts used in this example is available as files:

```
/star/examples/sg12/submean.cl
/star/examples/sg12/background.csh
```

If you intend to work through the example you should copy these files to your IRAF directory. The IRAF OIF image `ccdframe`, which was used in the previous examples, should also be available.

File `submean.cl` is a script of IRAF commands. It is shown in Figure 1. It calculates the mean value of all the pixels in an image and then generates a new image in which the mean value is subtracted from every pixel.

As with any unfamiliar programming language, your first reaction is probably one of trepidation as you wonder what is going on. No variables were declared at the start of this script although variables are used throughout the program for parameter passing. The `cl` has some built-in variables which do not need to be declared. There are three string variables named `s1` to `s3`, three real variables `x`, `y` and `z`, three integers `i`, `j` and `k`, three booleans and other more esoteric ones. Type '`lpar cl`' for a full list of the parameters available for immediate use. Other variables can be used but they need to be declared: see the `cl` manual[13] for information on how to do this. List-type parameters are useful, they are usually associated with files, and they allow you to use `fscan` and `fprint` for reading and writing to files. These functions are similar to the equivalent functions in the C language.

The command `imstat` is used to calculate the mean. Some IRAF tasks have output parameters which allow their results to be stored and subsequently accessed by another task. Unfortunately, `imstat` does not have any such output parameters. Instead, it is configured so that the only output that it generates is the mean value, without any annotation. This output is redirected to temporary file `temp.file` using the Unix-like output redirection capabilities of the IRAF `cl`.

The `fscan` function is used to read the value from the file and store it in variable `x`. The `imarith` task (from the `images` package) is then used to subtract the mean value from each pixel and create a new image `newframe`. Finally, `imstat` is reconfigured to display its default output and the statistics for image `newframe` displayed as a check that the script has worked correctly.

Quoted parameters are passed to the tasks as they are; the brackets around some of the parameter lists indicate that the `cl` language is meant to interpret these parameters as variables. Programming in `cl` can be tricky sometimes: putting '`echo=yes`' at the start of the script means that all commands are echoed to the terminal as they are executed, which can be useful for debugging. The file name '`ccdframe`' was of course hardwired into the script; a more typical application would operate on every file in a list, or loop until some condition is satisfied.

```
# IRAF script to calculate a mean and subtract
# it from an image.

# Load images package.
images

# Initialize the cl list variable.
list = " "

# Store a temporary file name in string s3.
s3 = "temp.file"

# Configure imstat to display only the mean
imstat.format=no
imstat.fields="mean"

# Redirect imstat output to temporary file.
imstat "ccdframe" > (s3)

# Associate temporary file with list variable.
list = (s3)

# Read the value in the file into parameter 'x'
i= fscan (list,x)

# Use imarith to subtract x from the image.
imarith ("ccdframe","-",x,"newframe")

# Reconfigure imstat and then look at statistics of
# new image to see if the subtraction worked.
imstat.format=yes
imstat.fields="image,npix,mean,stddev,min,max"

imstat ("newframe")

# Delete the temporary file.
delete ("temp.file", verify=no)

# Print an output message
print("Subtracted ",x,"to create file newframe.")
```

Figure 1: Example IRAF script

The `cl` can support loops of various types. See *A User's Introduction to the IRAF Command Language*[13] for more information.

The procedure to run the script is as follows.

1. First you must make the script known to IRAF. From the `cl>` prompt type:

   ```
   task $submean=home$submean.cl
   ```

   '`home`' is specifying that the script file resides in your root IRAF directory.

2. You can now simply run the script by typing:

   ```
   submean
   ```

   Output similar to the following should appear:

   ```
   #              IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
               newframe     98304  3.044E-5     308.3     -178.    31480.
   Subtracted 508.046 to create file newframe.
   ```

   You can check that image `newframe` has been created by typing `dir` and display it with SAOIMAGE by typing:

   ```
   display   newframe   1
   ```

3. Finally you should tidy-up by deleting the new image:

   ```
   imdelete   newframe
   ```

   Note that the script will not work correctly if `newframe` already exists, so you must delete it before running the script again.

14.1 Running a background job

It is possible to run IRAF scripts as background jobs. It is necessary to prepare a Unix shell script which starts IRAF and then runs the IRAF script. File `background.csh` is an example shell script. It is shown in Figure 2. This example invokes IRAF and uses it to run script `submean.cl`.

```
#!/bin/csh
# Make sure a C shell is used
# Run IRAF from the UNIX shell, taking parameters
# up until the word END.
cl  <<END  >log.file
images
task \$submean = home\$submean.cl
submean
logout
END
```

Figure 2: Example shell script to run IRAF as a background job

The script first ensures that a C shell is used, if it is run as an executable shell script, then IRAF is started and fed with commands. Notice how the `$` signs need to be escaped in the task definition to stop the UNIX shell from interpreting them as shell variables. The output is written to file `log.file`. There is further information about shell scripts in SC/4: *C-shell Cookbook*[6]).

The script is run from the Unix shell prompt. Simply type:

```
background.csh &
```

Output will be written to file `log.file`. Remember that script `submean.cl` will only run successfully if image `newframe` does not already exist.

A Obtaining and Installing IRAF

This appendix gives a few brief notes about obtaining and installing IRAF. A typical Starlink user is unlikely to be able to carry out a complete IRAF installation. The assistance of your site manager is likely to be required for at least some of the stages and he will probably carry out the complete installation.

IRAF is available for all the variants of Unix currently supported by Starlink (and on various other operating systems). A basic IRAF installation, without any external packages, requires about 100 Mbytes of disk space, though the actual amount varies somewhat between the different variants of Unix. However, about half this space can, if necessary, be reclaimed by deleting source files once the installation is complete.

A copy of IRAF can be retrieved by anonymous ftp from NOAO or the Starlink mirror. The IRAF Web pages (see Section 2) give the necessary details. Alternatively the NOAO can send you a copy on tape or CD-ROM, though a small charge is made. Again see the Web pages for details. There is a separate distribution of IRAF for each version of Unix supported and there is no compilation involved in making the installation (though source is supplied).

Before installing IRAF you should obtain and read the following documents:

1. The IRAF installation guide,
2. The IRAF management guide,
3. The `README` file for your particular machine,
4. The `REGISTRATION` file.

All these documents can be retrieved from the IRAF Web pages. There are different versions of the first three documents corresponding to each version of Unix supported and you should ensure that you are reading the correct version for the system on which you intend to install IRAF.

The `REGISTRATION` file is important as it contains a form to be filled in and returned to NOAO by e-mail, so that you can be added to the mailing list for the *IRAF Newsletter* (see Section 3.3).

Acknowledgements

We are grateful to Dave Bowen for advice and assistance with using IRAF and useful comments on the draft version of the document. Nigel Hambly also gave several helpful tips on using IRAF. Martin Bly, Alan Chipperfield and Rodney Warren-Smith provided useful comments on the draft.

Any mistakes, of course, are our own.

References

[1] J. Barnes, 1993, *A Beginner's Guide to Using IRAF* (National Optical Astronomy Observatories: Tucson). File: `beguide.ps.Z`. See Section 3 for details of obtaining IRAF documents.

[2] A.J. Chipperfield, 1997, SSN/35.3: *IRAFSTAR – The IRAF/Starlink Inter-operability Infrastructure* (Starlink).

[3] A.J. Chipperfield, 1997, SUN/217.2: *Running Starlink Applications from IRAF CL* (Starlink).

[4] A.J. Chipperfield, 1997, SUN/221.1: *KAPPA for IRAF* (Starlink).

[5] M.J. Currie, 1998, SUN/95.12: *KAPPA – Kernel Application Package* (Starlink).

[6] M.J. Currie, 1998, SC/4.2 *C-shell Cookbook* (Starlink).

[7] M.J. Currie, G.J. Privett, A.J.Chipperfield, D.S. Berry and A.C. Davenhall, 1998, SUN/55.8: *CONVERT – A Format-conversion Package* (Starlink).

[8] P.W. Draper, 1998, SUN/214.3: *GAIA — Graphical Astronomy and mage Analysis Tool* (Starlink).

[9] D.C. Kay and J.R. Levine, 1995, *Graphics File Formats*, second edition (Windcrest/McGraw-Hill: New York). See in particular Chapter 18, pp235-244.

[10] E. Krol, 1992, *The Whole Internet User's Guide and Catalog* (O'Reilly and Associates Inc: Sebastopol, California).

[11] R. Morris and G.J. Privett, 1966, SUN/166.4: *SAOIMAGE – Astronomical Image Display* (Starlink).

[12] J.W. Palmer, 1996, SUN/207.1: *AIPS – Astronomical Image Processing System* (Starlink).

[13] P.M.B. Shames and D. Tody, 1986, *A User's Introduction to the IRAF Command Language*, Version 2.3, (National Optical Astronomy Observatories: Tucson). File: `cluser.ps.Z`. See Section 3 for details of obtaining IRAF documents.

[14] K.T. Shortridge, H. Meyerdierks, M.J. Currie, M.J. Clayton, J. Lockley, A.C. Charles, A.C. Davenhall and M.B. Taylor, 1998, SUN/86.16: *FIGARO – A General Data Reduction System* (Starlink).

[15] B.J. Thomas, 1995, *The Internet for Scientists and Engineers* (SPIE – The International Society for Optical Engineering: Bellingham, Washington).

[16] R.F. Warren-Smith, 1998, SUN/33.5: *NDF – Routines for Accessing the Extensible N-Dimensional Data Format* (Starlink).

[17] J.V. Wall, 1979, 'Practical Statistics for Astronomers', *Q. J. R. Astron. Soc*, **20**, pp138-152.

[18] D.C. Wells, E.W. Greisen, and R.H. Harten, 1981, *Astron. Astrophys. Suppl*, **44**, pp363-370.