An Astronomer's Toolbox

Nicole P. Vogt

20 August, 2002

First Section

1. Getting information
   - People, whom to ask what
   - Library resources
   - Internet
2. General UNIX issues
   - UNIX at NMSU
   - Some UNIX commands
   - Printing and mailing
3. Editors
   - `emacs`
   - `vim`, `sed` and others
4. Perl
   - What and why
   - Using Perl

Second Section

1. Formatting text
   - LaTeX
   - Computer typesetting and the case for LaTeX
   - What it can do for you
2. Figures and plots
   - Production, `sm` etc.
   - Inclusion
3. Programming languages
   - C/C++
   - Fortran77/Fortran90
   - Debugging
4. Further scientific tools
   - Mathematica and Maple
   - Numerical Recipes
   - Iraf

1 Introduction

This handout is designed to provide you with enough information to send you off in the right direction to find most of what you need for your studies at NMSU. Some of it may even be of use later! The primary goal of this document is to be a reasonably useful guide to the computer system for the novice computer user, and to inform the seasoned UNIX guru about the tools available with the local system. It will not teach you how to use any program or programming language in detail, nor will it tell you all the neat and "cool" tricks one can do with various programs. Instead, you are expected to do some self-study and make use of other resources around the university, both human and otherwise.

This material was developed for a graduate-level orientation course at the Institute of Astronomy, Cambridge, UK. Before presenting an overview, I would like to thank Jarle Brinchmann and Melvyn Davies, who wrote the previous versions of these notes, and Bob Abraham and Vernon Bailey for useful suggestions.

2 Information resources

2.1 People

NMSU has a computer support department, which administers the Sun workstations within the astronomy department. These are contacted by mailing wsc@nmsu.edu (just type `mail wsc@nmsu.ed`). The PCs are supported in house, primarily by Jon Holtzman (holtz@nmsu.edu) and Joni Johnson (jojohnso@nmsu.edu).

These people are trained professionals, but for the more mundane tasks and problems, your immediate colleagues are equally important sources of knowledge. Don't feel too shy about asking other people how to do various tasks, many others have probably done the same before and asking their advice can save you large amounts of time (not to speak about the frustration you save yourself ⌣).

2.2 Written resources

2.2.1 Books

You might find it convenient to buy a book on the programming language you end up using, but don't go out and buy anything until you have perused the library, looked for books in the libraries and talked to other people that use the same language, computer that is. There is a wealth of poor books out there, and many of the good ones are not written with scientists in mind.

2.2.2 Computer files

Another important source of written information is the UNIX manual pages. These are specially formatted text files that are conveniently viewed using the `man`-program. Simply typing `man <command>` will show you information about that command. In particular `man man` gives you information about the `man` program itself. If you prefer a point-and-click interface you might want to try `xman`. Beware that these manual pages sometimes can be rather obscure (the standard example being the manual pages to the command `find`). They usually adhere to a structure with first a description of the command, then a listing of the options[1] the program accepts, an explanation of the usage (which usually merits repeated reading), a list of environment variables (see later) supported by the program and finally some examples of use.

2.3 The Internet, and why it might be a Good Thing

The Internet will most likely prove very useful during your research. The major programs used for interacting with the net at NMSU are `netscape` and `KDE Konqueror` (Internet Explorer is not used much here). The accounts are set up so that you will see the NMSU Astronomy Department Home Page on start-up. You reach other web sites either by explicitly using the open command with a particular site's address – its URL – or by clicking on one of the underlined options on a page. We suggest you simply explore for yourself.

Some places are immediately worth knowing about though. The Astrophysical Data Services (ADS) is really useful for finding references, and the abstracts, of papers written by a certain person, or on a certain topic. Astro-Ph is a clearing house for new papers; people submit manuscripts to it as eprints (electronic preprints), so looking at it is a good way to keep up with new work. The Users Guide to Computing is also worth a look. It contains a lot of information, that is useful for those people new to our computer system, and it is also a good reference for seasoned users. A useful reference on UNIX, can be found at

http://unixhelp.ed.ac.uk

You'll also need to know how to search for sites containing information on a particular topic. You can use **Google** (*http://www.google.com*), **Lycos** (*http://www.lycos.com*), **Yahoo** (*http://www.yahoo.com*) or

[1] In UNIX jargon options are usually a letter preceded by a dash: `-x`, though some programs, in particular GNU programs, might have longer option names.

Altavista (*http://www.altavista.com*), among others. Once you're reached any of these pages, simply give it the keywords to search by.

As you find useful places on the web, add them to your list of bookmarks. This list can be edited and restructured into hierarchies, something which is very useful as soon as you reach $> 30 - 40$ bookmarks.

3 General Unix issues

As you probably have gathered by now, most of the computers at NMSU run the UNIX operating system. Most computers run Linux 7.2 [2] and are PCs, but a few SunOS machines run Solaris.

You usually interact with machines here using the X-windows system. This is the underlying graphical window handling system, and on top on that is another program called the *window manager* which deals with your interaction with windows etc. The window manager of choice here is `KDE`.

3.1 A basic set of Unix commands

There are many commands in UNIX and they often have a large set of options to modify their behavior. To get an overview of this one must, as mentioned above, use the `man` command. The introductory leaflet will get you a bit more acquainted with the intricacies of UNIX. Here I will only list a selected set of useful commands. You might not use all of these, and you most certainly are going to use more than these, but this is a good start. If you are not already familiar with these commands, I suggest you go through the list using `man`.

System utilities

- `man` — Access online manual pages for UNIX command
- `ls` — List files
- `cp` — Copy files
- `mv` — Move files
- `rm` — Remove files
- `diff` — Compare files
- `cd` — Change directories
- `mkdir` — Create a directory
- `rmdir` — Remove a directory
- `chmod` — Change the ownership of files
- `more` — View textfile
- `alias` — Define shortcuts
- `ln` — Create soft links

Printing

- `lp` — Print a file
- `mpage` — Print several pages per physical page
- `a2ps` — Similar, but for ASCII files
- `lprm` — Remove print requests

[2] Strictly speaking Linux is not an operating system. It is an "operating environment" which includes the Linux OS and various other tools. The operating system is thus RedHat Linux release 7.2.

- `lpstat` — Check the status of your print requests
- `lpq` — Check the status of the printer queue

System information

- `ps` — Get the PID of your job
- `kill` — Use `kill -9 pid` to forcefully stop a process with PID `pid`
- `nice` — Lower the priority of a job
- `top` — Check how much resources your program occupies
- `who` — Check who is logged on
- `finger` — Intrude on someone's privacy (find out whether someone has read their mail or whether they are logged in)

Get out of town!

- `ssh` — Log onto another machine
- `sftp` — Fetch files from another machine
- `scp` — Copy files from another machine
- `logout` — Get out of there

Compilers

- `f77` — Fortran 77
- `f90` — Fortran 90
- `gcc` — Gnu C

Miscellaneous

- `sed` — Replace stuff in a file with some new stuff
- `awk` — Print columns in files

You should read through the Users Guide to Computing

http://ganymede.nmsu.edu/computing/notes

This gives information you ought to know about the system, and is a very useful reference later. In particular you are well advised to read the rules about backing up your own work.

3.2 Printing and mailing

3.2.1 Printing

You will find that you accumulate a lot of paper quite quickly, and it is quite easy to lose track of which version was printed out when. To save paper and help you keep track of your printouts, we recommend you use the two commands `a2ps` and `mpage`, both of which produce output with basic information about the printing, e.g. the name of the file and the date printed. `a2ps` prints text files, two pages per sheet by default. This is quite often used for program listings. The result of running `a2ps` will be written to the standard output, which means that you ought to redirect the output[3]. To print the file *prog.f* on the printer `down` you type

[3] If you wondered about two expressions in this sentence, you probably ought to read an introductory guide to UNIX

```
a2ps prog.f | lp -d down
```

The program `mpage` is quite similar, but works with Postscript files. By default this command will print four pages per sheet, but you can also make it print two or eight. To print the file *paper.ps* with two pages per sheet on `down` you just type

```
mpage -2 -Pdown paper.ps
```

Note that there must be *no* space between `-P` and `down`.

3.2.2 Mailing

You will be sending and receiving an awful lot of emails in the coming years. There are many mail programs in existence, but unless you really know what you are doing you might want to stick to `pine`. This is a reasonably easy-to-use mail program that you will find on many UNIX machines; it also has the advantage that many people at the university are using it so it is easy to get help. To start it you just type `pine` and follow the instructions.

You might in some cases find it more convenient to *send* mail to people using the bog-standard UNIX `mail` program.

Mail is typically something you are better off not experimenting with, in particular, do *not* mix two mail readers! That will cause you all kind of headaches, and even though your mail might not be lost you can have a lot of hassle finding it again, and in unfortunate situations you might even lose mail.

4 Editors

The questions of what editor to use is a sensitive one in UNIX circles, and it is not by chance that there is a newsgroup called `alt.religion.emacs` ... The only thing all seem to agree on is that one should not use Microsoft Word. We strongly recommend that you do not use MSWord if you can avoid it; it is not available on the UNIX machines. (If you need to import an MSOffice document, it can be read and manipulated with `OpenOffice` on the Linux boxes or with `StarOffice` on the Suns.)

I would suggest you use either `emacs`, `vi`, `vim` or `nedit`. Personally I use `emacs` or `xemacs` and would recommend this to all newcomers, but here is a quick run through the different programs

vi/vim stands for VIsual editor, and it is the standard UNIX editor[4]. As such it is widely used, and many old timers institute use `vi`. The advantage of `vi` is that whatever UNIX system you use, there will be a version of `vi` there, it is also compact and quick, much quicker to start up than for instance Emacs or `nedit`. The downside is that it is difficult to learn, more so than the next two editors. But if you want to check it out, you might want to have a look at the resources available at the Yahoo Vi pages

http://www.yahoo.co.uk/Computers_and_Internet/Software/Text_Editors/vi

If you are not going to use it, you only need to know that you get out of it by typing `:q` or `:q!` to avoid saving the file.

nedit is a small and compact editor with a fairly good selection of modes for support of programming in various programming languages. It is started by typing `nedit &`. Then set out using the menus. In particular you might want to look under `Preferences` where you can set the current mode and turn on and off syntax highlighting. Note however that few people use this editor here, so you might be a bit on your own. A useful resource is the `ne` homepage

http://www.emerson.emory.edu/services/editors/ne/Top.html

emacs is the choice of the GNU generation[5]. It is a massive program with extensive support for various

[4]Well, strictly speaking the standard text editor is `ed`, but you should never ever have to use this, and count yourself lucky indeed not to do so!

[5]To find out what GNU stands for, start `emacs` and type `C-h F` (where `C-h` means press the control key and h simultaneously)

programming languages and plenty of opportunities for time-wasting. To start `emacs` just type `emacs &`; for initial use you can probably get around by just using the menus. Note that you do not need to (and should avoid) start a separate `emacs` when you have to edit a new file. `emacs` keeps your files in buffers which can be accessed through the buffer menu. For an introduction to Emacs type `C-h t` inside Emacs to enter the Emacs tutorial. You can also get information from the Web, try for instance

http://www.geek-girl.com/emacs/emacs.html

You will also meet up with another editor if you use Pine to read email, namely `pico`. Even though this is fine for email writing, there is no compelling reason to use this editor for anything else.

5 Personal adjustments

Most people wish to adjust the working environment on their computer after the first days have gone past. This can take you three months and never leave you satisfied, or it can mean getting a different colour for the screen background. Since people use so many different programs, have so different tastes (though pink writing on green is ugly and bad for you no matter what you say), I will not go into this in any detail. Instead I will list the files of interest, from the most important, that most people will edit, to more obscure files that you might want to edit at some point.

The following files are (or should be put) in your home directory. Since they start with a dot, you will not see them when you type `ls`. To see such *dotfiles* (as they are called), you have to add the option `-a` (`ls -a`).

.mytcshrc This is the most important file you have in your home directory[6]. This file normally contains "aliases" and various environment variables that you have inserted yourself. If someone tells you to put something in your `.tcshrc`-file, then this is the file he or she means[7].

.xsession If you log into an X-terminal, then this file is read and any commands in this file will be executed at startup. You typically do this if you want to start some programs automatically every time you log in. Beware that if you screw up things here, you might need help to be able to log in again.

.xinitrc This is has the same function as *.xsession*, but it is executed when you don't log in via an X-terminal. The two files are normally identical.

.Xdefaults While we are on the topic of starting programs under X, we should mention this file, which contains the basic visual defaults for various programs. So if you for instance wanted Emacs to always start up with a green background and purple text, you would put the following into your *.Xdefaults*:

```
#ifdef COLOR
  Emacs*background:               green
  Emacs*foreground:               purple
#endif
```

To find out what you actually can put in, and what the syntax is, you should either read the man pages for the relevant program or look at someone else's *.Xdefaults*.

.emacs If you use Emacs to edit text, you might want to modify its behavior at time. This is then the file you have to put such modifications into. However, this is usually not necessary, because Emacs has a built-in customizer. Opinions about it might vary, but it is a reasonably easy way to adjust some of Emacs' behavior. You start it by pressing *Esc-x customize*.

[6] If you are not running `tcsh` you know what you are doing I hope

[7] You also have a *.tcshrc* file, but you should avoid editing this unless you really know what you do. If that's the case then you probably want to have a look at *.tcshrc_solaris* and *.tcshrc_linux* as well if they exist

.addressbook If you use *pine* to read mail, this is where your address book is saved.

.plan This is the file shown whenever someone tries to `finger` you.

.signature This is where you put the signature you want to append to all outgoing email. Large potential for nerd-points here...

There are plenty of chances to waste time here, but you can also end up optimizing your use of the computers.

6 Perl

6.1 What and why

In many situations you want to do some quick calculation, or to modify a text file very simply. Such tasks does not require a fully-blown programming language such as C or Fortran. They are often referred to as *scripting tasks* because they are often solved using a *script*.

There are several tools available for simple scripting tasks in UNIX. Those of you already familiar with UNIX should be familiar with shell-scripts and `grep`, and possibly `awk` and `sed`. These tasks are fairly simple to use (with the possible exception of `sed`) and are useful for small everyday tasks. But for larger applications, and if you need to learn something from scratch, **perl** is probably the best alternative. Perl is useful for text manipulation, handling of databases and simple plotting and a score of other tasks, that you probably will never need. It is also one of the most popular scripting languages on the Web.

There are two books for those wanting to learn perl: The Llama book ("Learning Perl") and the Camel Book ("Programming Perl") [8]. Reference information is available on the man pages (`man perl`).

6.2 Using Perl to manage catalogues and text

One of the most useful aspects of Perl is how well suited it is to deal with text files. Here I will quickly cover how one would read in a file and interchange the columns. Type the following text into the editor of your choice and save it is as *testperl.pl*. Type `chmod a+rx` *testperl.pl* to make the file executable. Then make a short file with two columns, mix text and numbers if you like, save the result in a file *dum.dat*. You can then test the perl script by typing: `testperl.pl` *dum.dat* and look at the result.

```
#!/usr/local/bin/perl
#
# The line above requests that /usr/local/bin/perl should be used to
# run the following script.
#
# Everything after a hash mark (except the construction on the first
# line) is considered a comment and ignored by  Perl.
#
while (<>) {  # Read every line in the file specified
   chop;      # Remove carriage return at the end of the line

#  If the line start (^) with zero or more (*) spaces (\s)
#  and then a word character (\w) we will interchange the rows.
   if ($_ =~ m/^\s*\w/) {
     ($x, $y) = split(' ', $_) # The line is put into the variable $_
                               # The split command splits each line at
                               # whitespace.
        print "$y $x\n";  # This will print the columns interchanged
   } else {
```

[8] There is also a Puma book for advanced Perl programming, but if you need that you are probably not reading this anyway

```
    print "$_\n";
}
```

OK, now for a little bit of explanation in addition to the comments above. The `<>` command is a general input operator which reads a line from the standard input and puts the line into the special variable `$_`. The `;` at the end of each line has to be there, as it designates end of command. The `if` statement compares the line (`$_`) with a *regular expression*. If the line starts with any number of spaces and then a letter or digit or underscore, the columns will be interchanged, otherwise the line will be unchanged (in case you had comments in the file starting with `#`). If you want to know more about regular expressions, type `man perlre`.

7 Writing formatted text

In astronomy people do *not* use MSWord, as mentioned previously. Instead we use an editor and LaTeX. LaTeX is a "language" for computer typesetting, something which means that the output is of very high quality. The downside is that most people find getting started with LaTeX quite a chore. But once you have learned to use it, you can produce documents very quickly. This is what you want when the final days of your thesis are coming up and you still have 100 pages to write, so learn LaTeX as soon as possible! Good places to learn about LaTeX are Leslie Lamport's User Guide and "The LaTeX Companion". The former introduces LaTeX and the latter shows you how to solve all sorts of difficult problems.

7.1 So what is it then?

Basically LaTeX is a typesetting program. You write your text, using the editor of your choice, inserting commands to indicate the sections, titles, mathematical formula. This file is the source file for your document. To format your document you run LaTeX on it, as e.g. `latex myfile.tex`. BTW. it is a very good idea to always call your LaTeX files `<something>.tex`. After `latex` has finished (assuming there were no syntax errors in your document), you will have a file called *myfile.dvi* which you can look at using `xdvi` — by typing `xdvi myfile.dvi &` — or convert into a postscript file with `dvips myfile.dvi`. The postscript file can then be printed out with `lp -d` *`down`* `myfile.ps`. This document, with extensive comments where new concepts are introduced, can be found in `toolbox.tex`.

LaTeX is particularly good at doing mathematics, and this is one of the major reasons for using it. However it is designed more for the average scientist than for the GR or QM expert who uses very complicated maths. If you find that you have troubles getting what you want, you ought to try a) AMSTeX which is a version of LaTeX developed by the American Mathematical Society and which can be included into LaTeX documents by including the line `\usepackage{amstex}` before `begin{document}`, or you might need to look in the TeX-book, but if you are that desperate you probably ought to ask people around you first ...

You will find that there are two different types of LaTeX available out there. The newer version is usually referred to as LaTeX2ϵ and this is the version I recommend you use when you are not writing papers as it is more stable and has more features. On the other hand, if you are going to write papers you will most likely have to use either *AASTeX* which is used for ApJ and some other journals, or *MN* which is used for Monthly Notices. These styles require you to use the older style. Quickly summarized, the difference is that the new style has a heading starting with `\documentclass` and uses `\usepackage` to include extension packages, whereas the old one start with `\documentstyle` and has extension packages as arguments to this. There are also differences for the text-changing commands: In the new version you specify italic text as `\textit{abc}` whereas in the old version you use `{\it abc\/}`. These differences are annoying, but they are fairly minor, and development is going away from the old version so I think it is advisable to learn the new version now.

8 Figures

8.1 Production

Sooner or later you'll have two columns of stuff and you'll want to plot one against the other. Again you have a choice, for everyday use we suggest either `pgplot`, or `sm`. Both are relatively straight forward to use, and manuals describing their use are available. You will also find these at most astronomical institutions, though `sm` is a commercial product so you might keep that in mind.

Some of you might be familiar with IDL which also is available here. This is a very powerful data analysis language, in reality it is a full programming language. Again, this is a commercial package and you should not depend on IDL being available wherever you go. However, if you have the need for complicated plots, advanced graphics and need to do calculations intermixed with plotting, you might want to have a look at IDL. Matlab is a similar package, and equally powerful.

If you want to make diagrams, `xfig` is really useful. Just type in `xfig &` and see what you get. It's menu driven and worth playing with to see what it can do. In addition we have a very powerful drawing program called `gimp`, which you start by typing in `gimp &`. There is an excellent Web site you can use for information

http://www.gimp.org

with a manual at

http://manual.gimp.org

This can do most of the presentation tasks you might want to do, and it is also very similar to Mac and PC based programs that you might have used before. If you want to render images in three dimensions, your best bet is to use `POVRay` — but it is very time consuming to design such an image, and I don't recommend it. You can find information at

http://www.povray.org

Also of use is `xv`. It's a useful program to let you view image files of various formats (Postscript – which is the format you send to a printer, as well as GIF, JPEG, and many others). You can grab a portion of the screen with it too. There is also a scanner available in the upstairs computer room.

To look at two dimensional data, you can use `saoimage` or `ximtool`. They are particularly good at making snazzy colour pictures for talks. See gurus for more information.

8.2 Inclusion

By now you should be an expert at producing figures and plots, the only thing left is to put them into documents. Since we have agreed that we should only be using LaTeX, the discussion here will focus on this.

The modern version of LaTeX has built-in support for graphics, if you want the whole truth you should have a look (with `xdvi`) at

/usr/local/teTeX/texmf/doc/latex/graphics/grfguide.dvi

but let me just say that what you need to get started is to use the figure environment and the command `includegraphics`, as in the following example which will include the Postscript file *dum.ps* into a LaTeX document and scale it so it takes up 50% of the width of the text

```
..

\usepackage{graphicx}
```

```
..
\begin{document}
..

\begin{figure}
   \includegraphics[width=0.5\textwidth]{dum.ps}
   \caption{An example figure}
\end{figure}

..
```

With the old type of LaTeX this would achieved using a package called `epsfig`, as follows:

```
\documentstyle[epsfig]{article}
..
\begin{figure}
   \epsfig{file=dum.ps,width=0.5\textwidth}
..
```

After a while you are going to want to place a figure at some particular place in your document. Be warned that this is not always easy in LaTeX, but a good first stop would be to have a look in the "LaTeX companion" which you can find in the library.

9 Compilers

A few brief words about the two main families of compilers. For those who don't know what a compiler is, it is, roughly speaking, a program which translates one or more files containing instructions from a programming language into an executable code.

9.1 C, C++

A language that is very well suited to system programming, maybe less so to numerical work, but many people love it. In addition to the regular C compiler (called `cc`) we have the GNU version (`gcc`). C++ is also available at least the GNU project C++ compiler (`g++`), ask the gurus if interested. Books describing C are around, but you may want to buy one. You are not short of choices, so ask around, and you might also find that the C Frequently Asked Questions (FAQ) is a very useful reference

http://www.eskimo.com/%7Escs/C-faq/top.html

Kernighan and Richie is good if you already know the language, but you might find it a bit dry (unless you are the kind of person who like Borel sets and 3j symbols) Typing `man cc` or `man gcc` will give you a list of the compiler options.

9.2 Fortran

Fortran is the programming language of choice in astronomy, it is certainly not the best programming language in the world, but it is well suited for numerical computing. Most people still use Fortran 77 which can be invoked as `f77`, but there is a trend towards using Fortran 90 (invoked as `f90`), which is a huge step forward in terms of programming style from Fortran 77. For those interested in learning about Fortran 90, there's a good book by Metcalf and Reid entitled "Understanding Fortran 90", by Oxford University Press. But there are other books around, again, have a look around. A useful net resource is the Fortran Market

http://www.fortran.com/fortran/market.html

9.3 Debuggers

If your program crashes or misbehaves you need to debug it, and to help you here there are several debuggers. The Sun workshop system has a built-in graphical debugger. This system can be started by typing `workshop &` on the command-line. There is also extensive on-line documentation.

The other major debuggers are `dbx` and `gdb`. `dbx` is the only one that supports Fortran 90 properly. Important points: you must compile your program with the "-g" option, and with all optimizers off. See gurus for more information, as required.

10 Further scientific tools

There are many other tools out there that you might find useful at some point. Here I have collected an overview of some programs you might find useful in your work. They all have extensive documentation, some in the library and some on-line.

10.1 Mathematica and Maple

If you end up doing a lot of analytical calculations, you are well advised to try out either Mathematica or Maple. They may or may not be available at NMSU, but are still worth mentioning. These packages are both well suited to do symbolic manipulations and also limited numerical calculations. As an example, here is how you integrate $1/(x^4+1)$ in the two packages

Maple: `int(1/(x^4+1), x);` Note the final semi-colon, this has to be there to get Maple to evaluate the expression.

Mathematica: `Integrate[1/(x^4+1), x]` Note that the function is capitalized, this is a feature of all Mathematica functions.

To start maple you type `maple` for a text interface or `xmaple` for a fancy graphics interface. Mathematica is started by typing `math`, again for a text interface and `mathematica` for the graphical interface. The graphical interfaces have powerful "notebook" features which enables you to save your work in a nicely formatted form for later perusal.

10.2 Numerical Recipes

There are many books on numerical methods/algorithms. Numerical Recipes written by Press et al gives a good review of a lot of stuff and is a good book to look at, especially if you're only ever going to look at one book. The routines are also available online, so don't type them in by hand if you plan to use them.

10.3 Iraf and data reduction

Most of you will at some point come face to face with observational data that have to be reduced (even if you mainly do theory). A common option is to use Iraf. Iraf is a heterogeneous collection of routines and packages for all sorts of tasks, but it is first of all a data reduction environment. It is also rather difficult to get started using it.

Well, starting it only requires a minor effort on your part. I suggest you create a directory called, say, *Iraf* and then run a program called `mkiraf` in this directory to set up some files required by Iraf. You are then ready to start Iraf by typing `cl`. So the process is as follows:

```
>mkdir Iraf
>cd Iraf
>mkiraf
>cl
 NOAO Sun/IRAF Revision 2.11 Tue Dec 23 23:22:19 MST 1997
```

```
 This is the EXPORT version of Sun/IRAF V2.11 for SunOS 4 and Solaris 2.6

 Welcome to IRAF.  To list the available commands, type ? or ??.  To get
 detailed information about a command, type ‘help command’.   To  run  a
 command  or  load  a  package,  type  its name.   Type  ‘bye’ to exit a
 package, or ‘logout’ to get out of the CL.   Type ‘news’  to  find  out
 what is new in the version of the system you are using.   The following
 commands or packages are currently defined:

   apropos     dataio.     fitsutil.   mscred.     phip.       spptools.
   ared.       dbms.       flair.      multired.   plot.       stsdas.
   aspec.      digiphotx.  focas.      nmisc.      proto.      system.
   color.      dimsum.     images.     noao.       rgo.        tables.
   crutil.     figaro.     language.   obsolete.   rvsao.      utilities.
   ctio.       finder.     lists.      phiirs.     softools.   xray.
cl>
```

To get help on various topic you use the Iraf command `help`, this works similarly to the UNIX command `man`. To get out of Iraf you type `logout`. When you have gotten that far and you feel like more windows on your screen you can start the image display program XImtool by typing `ximtool &` and start Iraf again. If you type `display dev$pix` on the Iraf command-line, you get a beautiful image in your image buffer.

If you want to get back to a previous command, you have to appreciate that Iraf saves you from the scourge of arrow-keys, if you actually want to get back to previous commands and not just want to write funny characters with your arrow keys you have to type “`e`” on the Iraf command-line (which incidentally is not something you want to type on the UNIX command line — if you do, press `q` to get out). But by now you really need to read something dedicated to Iraf, so I suggest you have a look at the Web pages:

http://iraf.noao.edu/docs/docmain.html

read whatever you find useful, but most importantly the beginner’s guide to Iraf. You will also find some introductory exercises to Iraf, these are helpful, but you should be aware that it is usually more efficient to have someone show you the basic steps.

That’s it. Good Luck!