Introduction to IRAF

Guillermo Damke (TA)
Steve Majewski (Course instructor)

ASTR 5110 – Fall 2011
University of Virginia

What is IRAF?

- Image Reduction an Analysis Facility,
- "a general purpose software system for the reduction and analysis of astronomical data."
- "IRAF is written and supported by the IRAF programming group at the National Optical Astronomy Observatories (NOAO) in Tucson, Arizona."
- Quotes from the IRAF main website: http://iraf.noao.edu
- However, you can find documentation and support at http://iraf.net .

- This site was started in early 2006 as a volunteer effort by the IRAF development team after NOAO decided to end community user support and development of IRAF.
- READ THE BEGINNER'S GUIDE TO IRAF: http://iraf.net/irafdocs/beguide/
- OTHER HELPFUL GUIDES: http://iraf.net/irafdocs/

IRAF basic structure

- IRAF is composed of *packages* and *tasks*.

They are ordered hierarchically, like a tree with branches and leaves.

- Each *package* contains either more *packages* or *tasks*.
- What you execute are the *tasks*.
- Imagine the tree again. Branches are *packages* and leaves are *tasks,* then:
    - A branch (package) may split into more smaller branches (packages), or
    - a branch (package) will end in many leaves (tasks).
- A given *package* groups related *packages* or *tasks*. For example, some packages are:
    - *imred*: Image reduction, contains many packages, and is located in the package *noao;*
    - *twodspec*: 2-d spectra, contains two packages inside;
    - *onedspec*: 1-d spectra, contains *tasks* inside.
- You always load the *packages* that you need, and then execute their *tasks*.
- Some *packages* are loaded by default.
- You can customize which packages are loaded by default in your *loginuser.cl* file.

Setting up IRAF at UVa

- IRAF stores some environment variables in a file called *login.cl*.
- Furthermore, it stores the options for any *task* in a directory named *uparm*.
- This file and directory are created at the initial setup by the *mkiraf* command.
- You can customize what IRAF loads at startup in a file called *loginuser.cl,* which has the format of *login.cl* file.
- Standard practice is to let local *mkiraf* command make *login.cl*, and you customize *loginuser.cl* version.
- IRAF will look for *login.cl* and *loginuser.cl* everytime you start IRAF.
- The usual place to set up IRAF -- which is where *login.cl, loginuser.cl* and *uparm* will live -- is in a directory named *iraf* in your home directory.
- You will see how to setup IRAF in the next slide.

Setting up IRAF at UVa

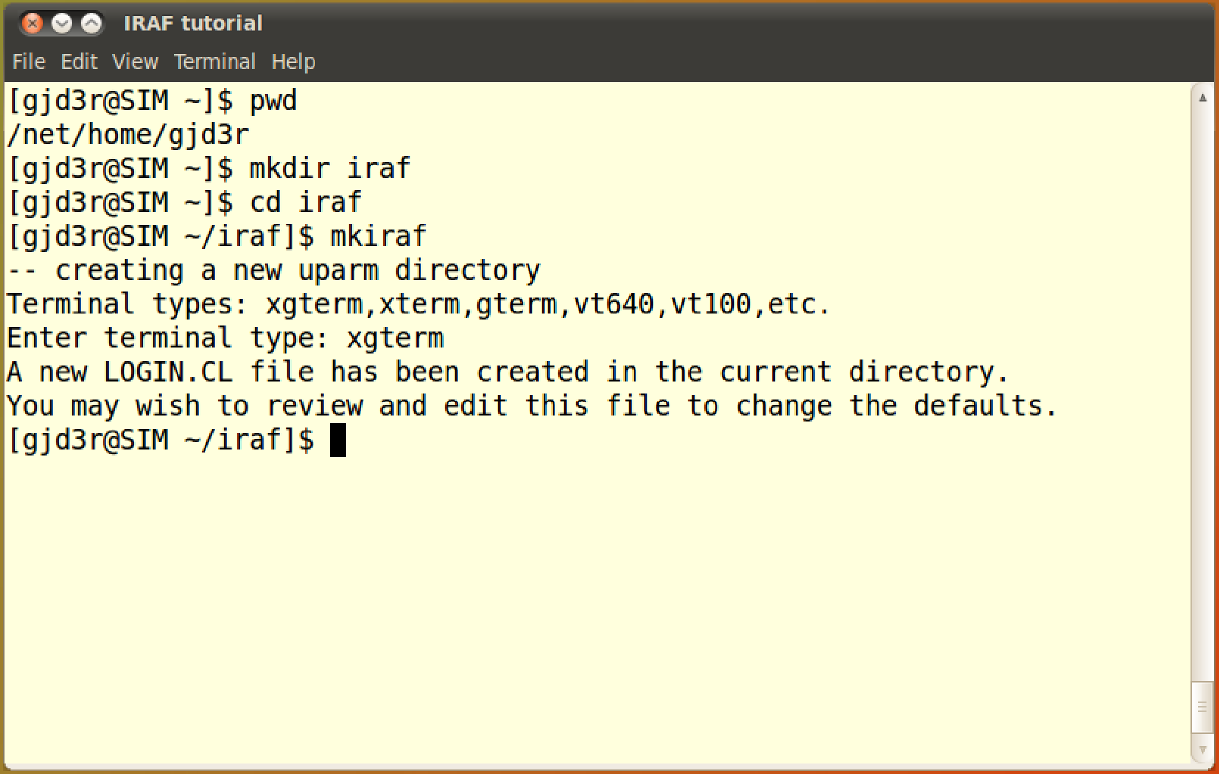
- First, you have to define some environment variables:
- Open a terminal, and add the following lines to your *.cshrc* or your *.cshrc.linux* file:

```
setenv iraf /astro/iraf/iraf/
setenv home $iraf
setenv host $iraf/unix/
setenv hlib $iraf/unix/hlib/
setenv hbin $iraf/unix/bin/
setenv IRAFARCH `$iraf/unix/hlib/irafarch.csh -actual`
```

- If you have problems, ask Howard Powell or Ricky Patterson.

Setting up IRAF at UVa

Now you can create your *iraf* directory, and run *mkiraf*.



```
[gjd3r@SIM ~]$ pwd
/net/home/gjd3r
[gjd3r@SIM ~]$ mkdir iraf
[gjd3r@SIM ~]$ cd iraf
[gjd3r@SIM ~/iraf]$ mkiraf
-- creating a new uparm directory
Terminal types: xgterm,xterm,gterm,vt640,vt100,etc.
Enter terminal type: xgterm
A new LOGIN.CL file has been created in the current directory.
You may wish to review and edit this file to change the defaults.
[gjd3r@SIM ~/iraf]$ 
```

You may need to edit your *login.cl* file. Uncomment *stdimage* and *imtype* variables.

set stdimage = imt800

Use: imt2048, imt4096, imt8192, etc.

set imtype = "fits"

Running IRAF: The xgterm Window

- IRAF uses a special type of terminal called *xgterm*.
- An *xgterm* can handle *text* and *graphics* (plots).
- In a terminal, open an *xgterm* by typing:
    - % xgterm &
- However, your TA prefers to open an *xgterm* with a scroll bar and bigger font size.

In this case, you can add options:

- % xgterm -sbr -fn 9x15 &

- You can find more options by typing:
    - % xgterm -help

Running IRAF: The Image Display

- Furthermore, IRAF typically needs an image display. You can choose either *DS9* or *Ximtool.*
- Open your image display **from the *xgterm* before** you launch IRAF. Type:
    - $ ds9 &

Or:

- $ ximtool &

Running IRAF: Starting IRAF

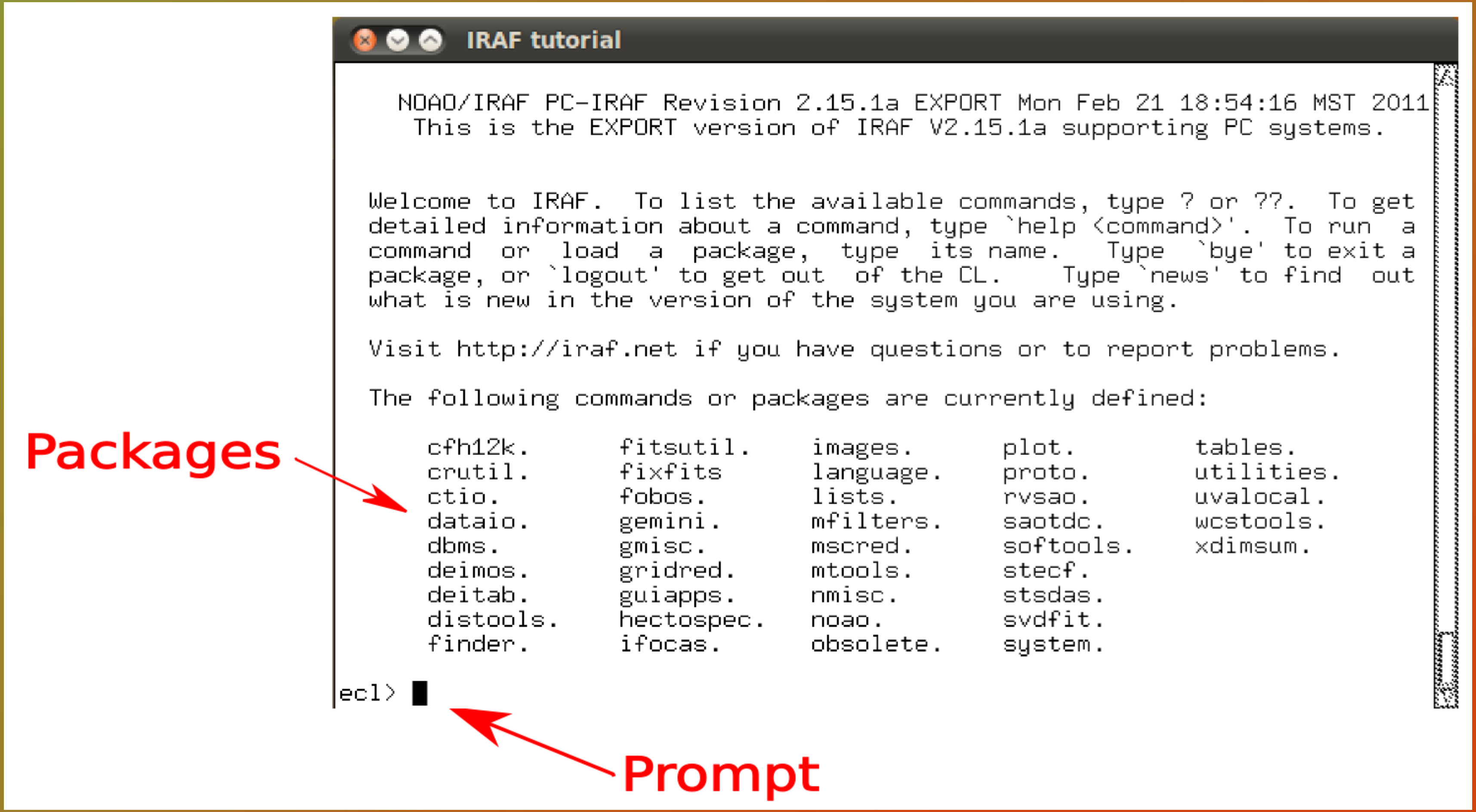
- From the *xgterm,* cd to the *iraf* directory -- where your login.cl file is -- and start IRAF by executing *ecl*.
- *ecl* stands for *enhanced-cl (enhanced command language)*.
- In the past, you would've run *cl*. The *ecl* is an improved version of the *cl.*

Read about the differences between *ecl* and *cl* in IRAF with the command:

```
ecl> page pkg$ecl/Notes.ecl
```

- The *ecl* and the *cl* handles the IRAF commands, but also allows you to use *Unix* commands, as:
  - cd
  - ls
  - pwd
  - copy (but use *imcopy* to copy images in IRAF),
  - rm (but use *imdelete* to delete images in IRAF),
  - rename (but use *imrename* to rename images in IRAF),
  - Etc.

Running IRAF

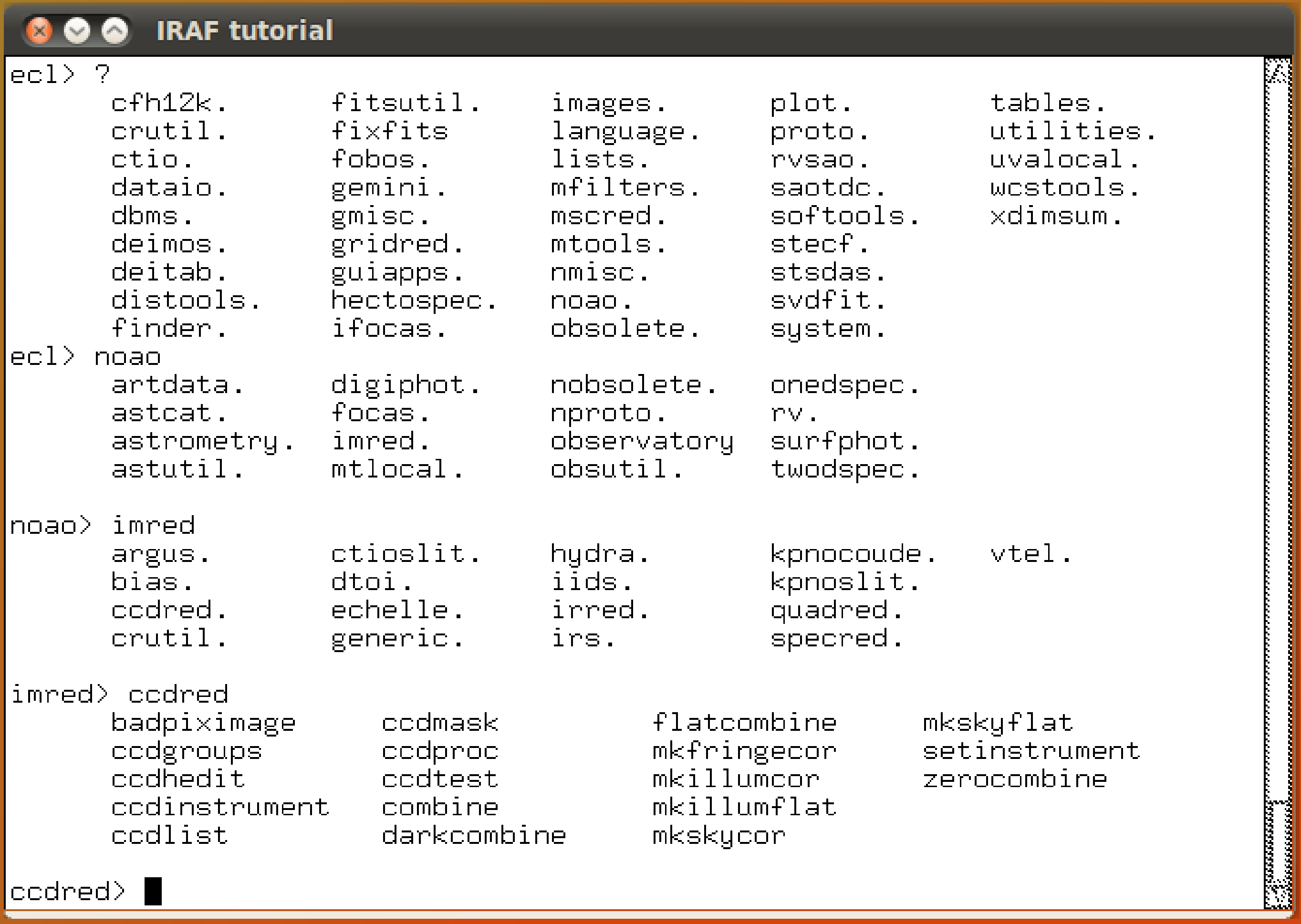


```
   NOAO/IRAF PC-IRAF Revision 2.15.1a EXPORT Mon Feb 21 18:54:16 MST 2011
    This is the EXPORT version of IRAF V2.15.1a supporting PC systems.


  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.

  Visit http://iraf.net if you have questions or to report problems.

  The following commands or packages are currently defined:

      cfh12k.       fitsutil.     images.       plot.         tables.
      crutil.       fixfits       language.     proto.        utilities.
      ctio.         fobos.        lists.        rvsao.        uvalocal.
      dataio.       gemini.       mfilters.     saotdc.       wcstools.
      dbms.         gmisc.        mscred.       softools.     xdimsum.
      deimos.       gridred.      mtools.       stecf.
      deitab.       guiapps.      nmisc.        stsdas.
      distools.     hectospec.    noao.         svdfit.
      finder.       ifocas.       obsolete.     system.

ecl> █
```

IRAF welcome window.
IRAF prints package names ending in a dot.
IRAF prints task names ending without a dot.

IRAF Packages

- As an example, we will load the *ccdred* package, used for ccd reductions.
- This package is inside the *imred* package.
- The package *imred* is inside the *noao* package.
- See the screenshot on the right:
  - We print the available packages by typing: ecl> *?*
  - We load the *noao* package. It contains many packages (names end with a dot).
  - Then we load the *imred* package. Again, it contains more packages.
  - Finally, we load the *ccdred* package. It contains *tasks*, and names do not end with a dot.
- To *unload* a package, type: ecl> *bye*

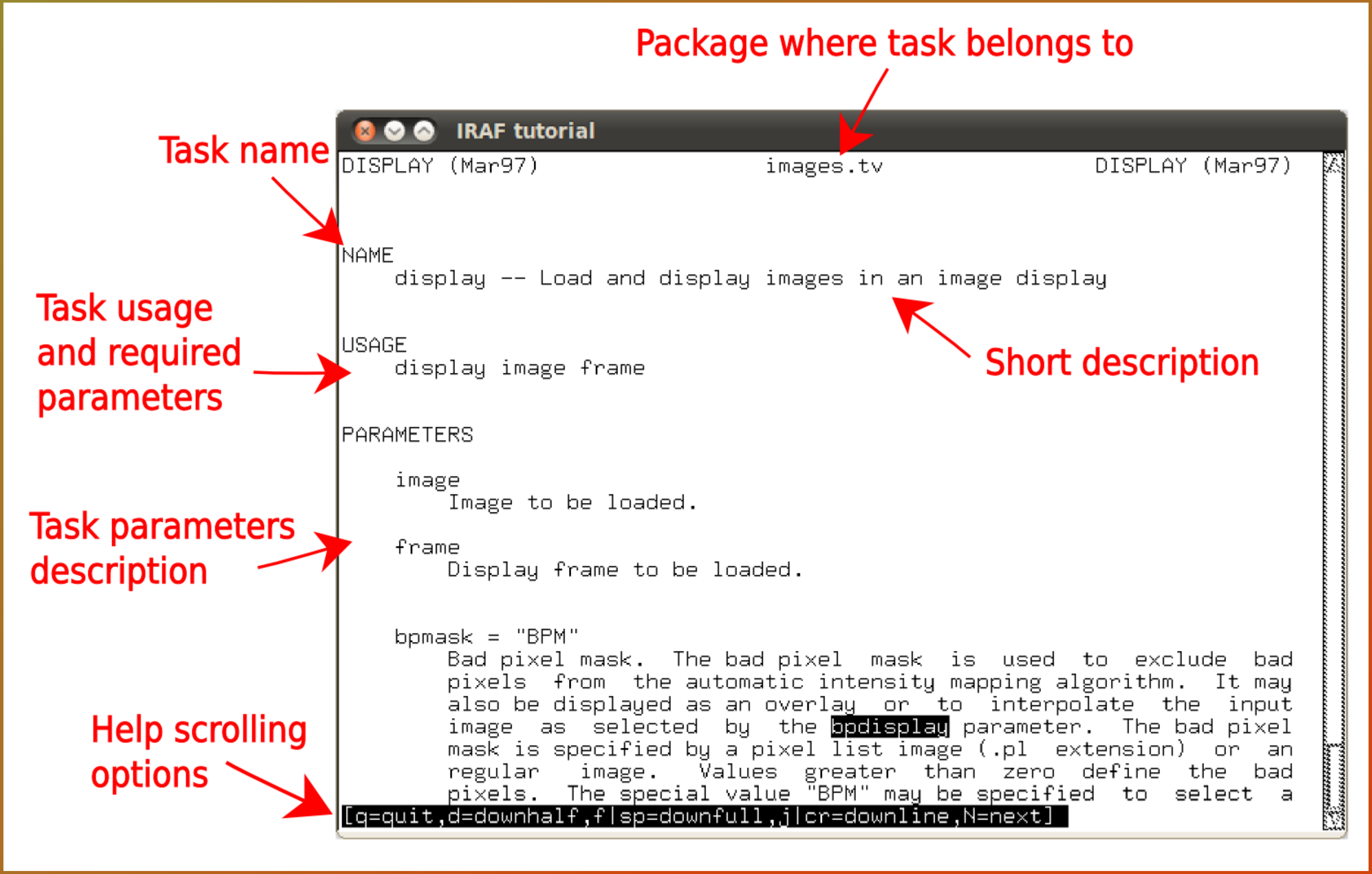

IRAF tutorial

```
ecl> ?
      cfh12k.       fitsutil.     images.       plot.         tables.
      crutil.       fixfits       language.     proto.        utilities.
      ctio.         fobos.        lists.        rvsao.        uvalocal.
      dataio.       gemini.       mfilters.     saotdc.       wcstools.
      dbms.         gmisc.        mscred.       softools.     xdimsum.
      deimos.       gridred.      mtools.       stecf.
      deitab.       guiapps.      nmisc.        stsdas.
      distools.     hectospec.    noao.         svdfit.
      finder.       ifocas.       obsolete.     system.
ecl> noao
      artdata.      digiphot.     nobsolete.    onedspec.
      astcat.       focas.        nproto.       rv.
      astrometry.   imred.        observatory   surfphot.
      astutil.      mtlocal.      obsutil.      twodspec.

noao> imred
      argus.        ctioslit.     hydra.        kpnocoude.    vtel.
      bias.         dtoi.         iids.         kpnoslit.
      ccdred.       echelle.      irred.        quadred.
      crutil.       generic.      irs.          specred.

imred> ccdred
      badpiximage      ccdmask          flatcombine      mkskyflat
      ccdgroups        ccdproc          mkfringecor      setinstrument
      ccdhedit         ccdtest          mkillumcor       zerocombine
      ccdinstrument    combine          mkillumflat
      ccdlist          darkcombine      mkskycor

ccdred> █
```

IRAF Tasks

- We will ilustrate how to work with tasks by using the task *display*.
- *Display* is loaded by default.
  - This task is used to visualize images in the image display.
- To see the help about a task, type:

ecl> help task_name


Package where task belongs to
Task name
Task usage and required parameters
Short description
Task parameters description
Help scrolling options
IRAF tutorial

```
DISPLAY (Mar97)              images.tv                  DISPLAY (Mar97)


NAME
    display -- Load and display images in an image display


USAGE
    display image frame


PARAMETERS

    image
        Image to be loaded.

    frame
        Display frame to be loaded.


    bpmask = "BPM"
        Bad pixel mask.  The bad pixel  mask  is  used  to  exclude  bad
        pixels  from  the automatic intensity mapping algorithm.  It may
        also be displayed as an overlay  or  to  interpolate  the  input
        image  as  selected  by  the bpdisplay parameter.  The bad pixel
        mask is specified by a pixel list image (.pl  extension)  or  an
        regular  image.  Values greater than zero define  the  bad
        pixels.  The special value "BPM" may be specified  to  select  a
[q=quit,d=downhalf,f|sp=downfull,j|cr=downline,N=next]
```

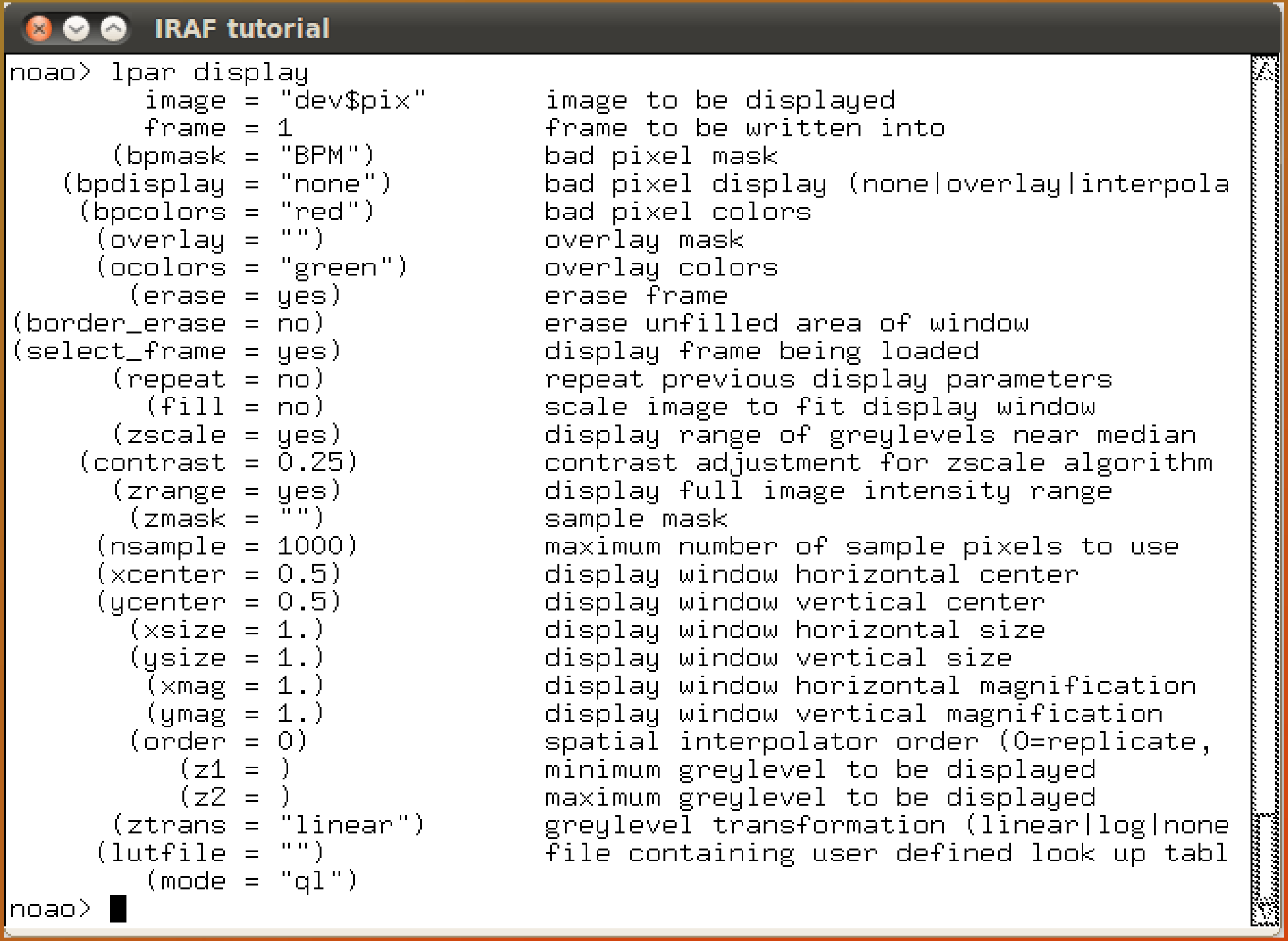
The *display* task help

IRAF Tasks: Parameters

- Use the command *lpar* to list the parameters of a task.

  ecl> *lpar task_name*

- *Required parameters*
  - Listed at beginning, without parentheses.
  - Must be specified each time a command is invoked.
  - Appear under *USAGE* in the task help.
- *Hidden parameters*
  - Listed in parentheses.
  - Not requested when task run.
  - Often defaults work well.
- Note that parameters may be:
  - String
  - yes | no (Boolean)
  - Float
  - Integer



```
noao> lpar display
        image = "dev$pix"       image to be displayed
        frame = 1               frame to be written into
     (bpmask = "BPM")           bad pixel mask
  (bpdisplay = "none")          bad pixel display (none|overlay|interpola
   (bpcolors = "red")           bad pixel colors
    (overlay = "")              overlay mask
    (ocolors = "green")         overlay colors
      (erase = yes)             erase frame
(border_erase = no)             erase unfilled area of window
(select_frame = yes)            display frame being loaded
     (repeat = no)              repeat previous display parameters
       (fill = no)              scale image to fit display window
     (zscale = yes)             display range of greylevels near median
   (contrast = 0.25)            contrast adjustment for zscale algorithm
     (zrange = yes)             display full image intensity range
      (zmask = "")              sample mask
    (nsample = 1000)            maximum number of sample pixels to use
    (xcenter = 0.5)             display window horizontal center
    (ycenter = 0.5)             display window vertical center
      (xsize = 1.)              display window horizontal size
      (ysize = 1.)              display window vertical size
       (xmag = 1.)              display window horizontal magnification
       (ymag = 1.)              display window vertical magnification
      (order = 0)               spatial interpolator order (0=replicate,
         (z1 = )                minimum greylevel to be displayed
         (z2 = )                maximum greylevel to be displayed
     (ztrans = "linear")        greylevel transformation (linear|log|none
    (lutfile = "")              file containing user defined look up tabl
       (mode = "ql")
noao> 
```

IRAF *lpar*

IRAF Tasks: Editing Parameters

- Use the command *epar* to modify the parameters of a task.

  ecl> *epar task_name*

- Scroll up and down with arrows.
- You can save the parameters by typing *:w*
- You can save and exit by typing *:q*

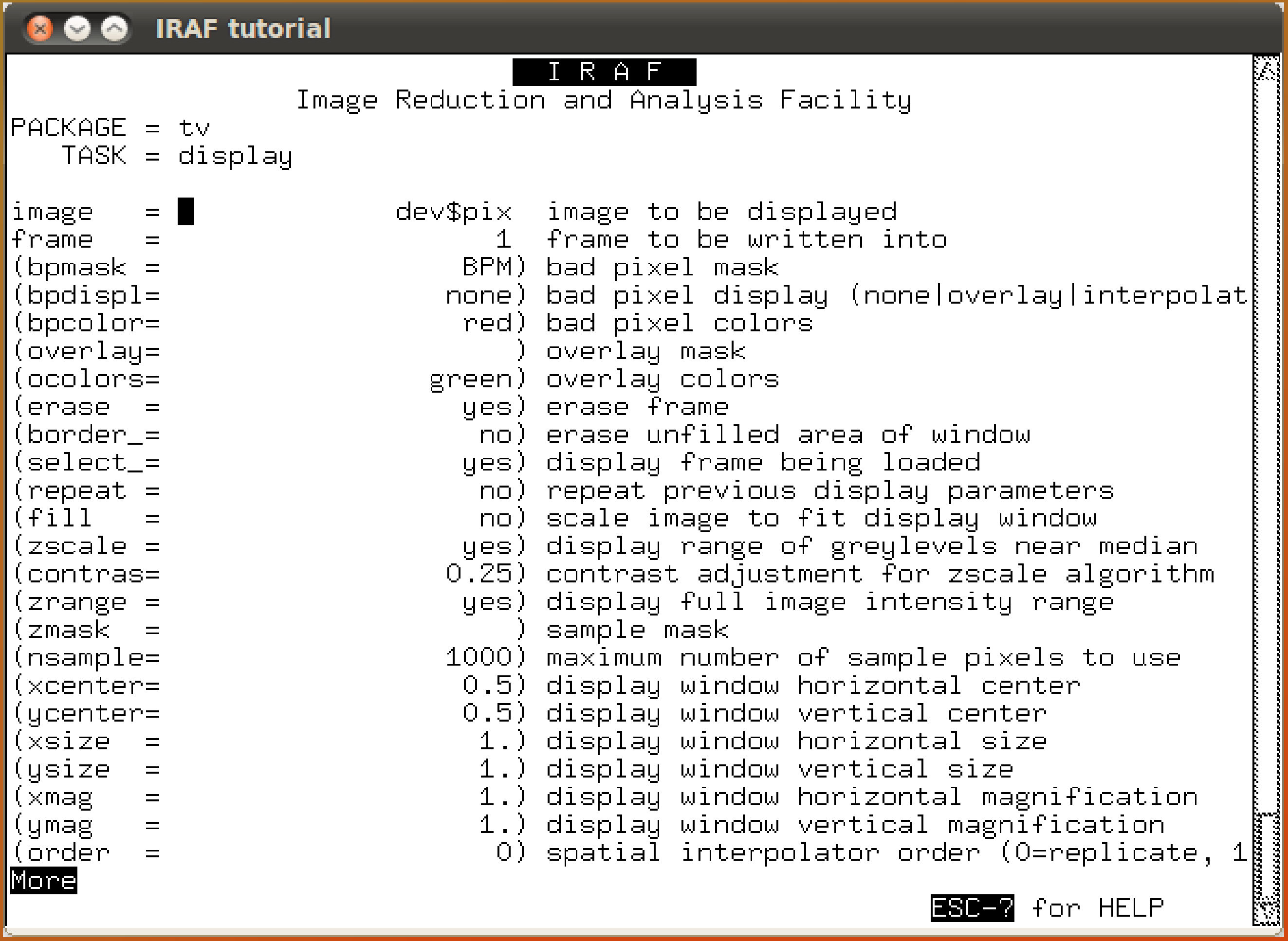
  or *:wq* or *ctrl+d*
- You can exit without save by typing *:q!*
- You can restore the default params. by exiting and typing:

  ecl> *unlearn task_name*

- You can execute a task from *epar* by typing *:go*

  (check DS9 now!)
- *TA Tip:* To modify a string, type:

  ctrl+a ctrl+u ctrl+l



```
                                     I R A F
                      Image Reduction and Analysis Facility
PACKAGE = tv
   TASK = display

image   = █                dev$pix  image to be displayed
frame   =                        1  frame to be written into
(bpmask =                      BPM) bad pixel mask
(bpdispl=                     none) bad pixel display (none|overlay|interpolat
(bpcolor=                      red) bad pixel colors
(overlay=                         ) overlay mask
(ocolors=                    green) overlay colors
(erase  =                      yes) erase frame
(border_=                       no) erase unfilled area of window
(select_=                      yes) display frame being loaded
(repeat =                       no) repeat previous display parameters
(fill   =                       no) scale image to fit display window
(zscale =                      yes) display range of greylevels near median
(contras=                     0.25) contrast adjustment for zscale algorithm
(zrange =                      yes) display full image intensity range
(zmask  =                         ) sample mask
(nsample=                     1000) maximum number of sample pixels to use
(xcenter=                      0.5) display window horizontal center
(ycenter=                      0.5) display window vertical center
(xsize  =                       1.) display window horizontal size
(ysize  =                       1.) display window vertical size
(xmag   =                       1.) display window horizontal magnification
(ymag   =                       1.) display window vertical magnification
(order  =                        0) spatial interpolator order (0=replicate, 1
More
                                                         ESC-? for HELP
```

IRAF *epar*

IRAF Task: Running from Command Line

•A second option to execute tasks is to type the name of the task and the required parameters on the ecl command line (1).

•If you don't enter required parameters, IRAF will ask you for them (2).

•You may even pass hidden parameters by their names (3):

- Boolean parameters are entered by *name+* (yes), or *name-* (no)

  Alternatively: *name=yes* or *name=no*

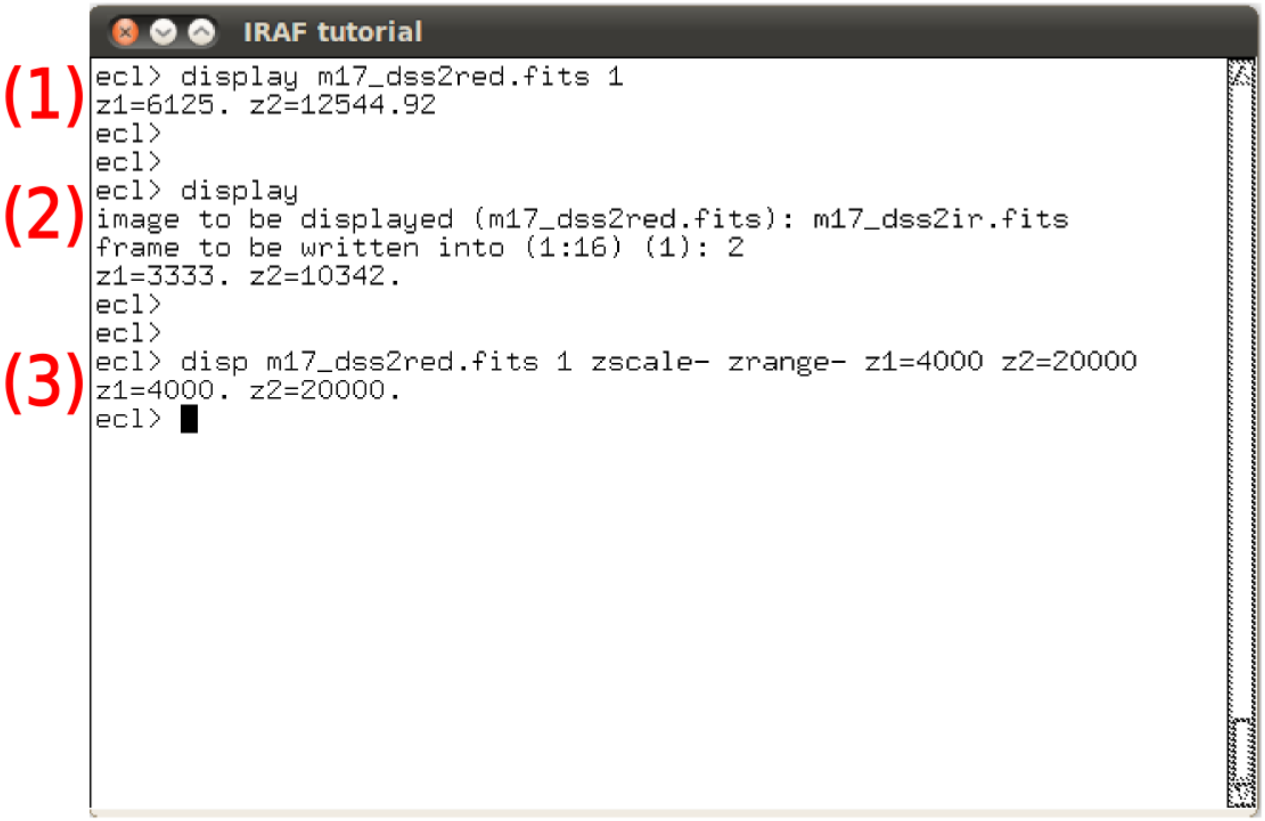
- Non-boolean parameters use the

  name =value format.

•*TA Tip:* IRAF will recognize the name of a task as soon as no other task begins with the same characters you type in.

For example, try typing *disp*, *lpar disp,* or *epar disp.*

See (3).



```
ecl> display m17_dss2red.fits 1
z1=6125. z2=12544.92
ecl>
ecl>
ecl> display
image to be displayed (m17_dss2red.fits): m17_dss2ir.fits
frame to be written into (1:16) (1): 2
z1=3333. z2=10342.
ecl>
ecl>
ecl> disp m17_dss2red.fits 1 zscale- zrange- z1=4000 z2=20000
z1=4000. z2=20000.
ecl> █
```

Header Tasks

- To list a header, use *imheader*.

  ecl> *imheader image_name*

- To list a whole header, use the *longheader = yes* parameter:

  ecl> *imheader image_name l+*

- To list specific keywords, use *hselect*.

  ecl> *hselect image_name $I,kwrd1,kwrd2,etc.*

  - $I means "image file name". Then hselect displays image_name, kwrd1, kwrd2, etc.
  - Note that there are no spaces between the keywords!

- To add, edit or delete header keywords, use *hedit*:

  ecl> *hedit images fields value*

  - Check *help hedit* for full description on usage.

Desiderata

- In IRAF, you can redirect stdout to a text file (>) (>> for appending), as in Unix. For example, to save hselect output to my_header_entries.txt:

  ecl> *hselect $I,kwrd1,kwrd2,etc. > my_header_entries.txt*

- Most IRAF input parameters accept *wildcards* as input, as in the Unix terminal. For example, to use *hselect* for all of our M17 FITS images:

  ecl> *hselect m17*.fits $I,kw1,kw2,etc. yes*

- Type "*e*" (without quotes) to see/edit the previous commands (same as scroll up).
  - Type *"e command"* to see/edit the previous call to *command.*
- Type *"flpr"* (without quotes) *twice* everytime a task is interrupted, or if IRAF behaves abnormally. The *flprchache* command clears the *process cache* where tasks live after they are executed the first time.
- Execute Unix commands that are not recognized or are amiguous to the *ecl* by preceding the command with the escape character *!* . For example, to run the Unix commands *hostname* or *whoami:*

  ecl> *!hostname*

  ecl> *!whoami*

Desiderata

- ***NEVER, NEVER, NEVER use ctrl+c in IRAF. It will kill all of IRAF!***
- Specify *image sections* with brackets.

  The format is *[x1:x2, y1:y2].*

  For example, to copy an image section of all the columns between line 100 and 200:

  ```
  ecl> imcopy image1.fits[*,100:200] image2.fits
  ```

  * means *all the lines or columns,* depending on where you place it.

- **Maybe the most useful "trick" is to use .txt lists as task input or output.**

  You can define lists of images (for example, your bias, flats, objects, etc.) and use them for task input or output. To do this:

  - Create an input list using the *ls* command and redirect with >.
  - You can copy this list for output using *cp*, and modify names quickly using *vi,* or other text editor (for example, to rename or copy images).
  - IRAF recognizes a list of files if you add the @ character at the beginning of the .txt file name. For example:

    ```
    ecl> hselect @my_files.txt I$,kwrd1,kwrd2,etc yes
    ```

The *imexamine* Task

- Imexamine is a task to "examine images using image display, plots, and text". This task works interactively.
- To run *imexamine,* type:

  ecl> *imexamine image_name*

  If you already displayed your image, just type *imexamine*
- When you run *imexamine*, you will see that:
  - The image is displayed in your image display.
  - The cursor in the image display changes to an open circle. This means that it is waiting for commands.
  - Many *imexamine* commands plot your data. In this case, a new window will open. (This type of plots is the reason why you must run IRAF in an *xgterm* instead of an *xterm*).
  - ***NEVER, NEVER, NEVER close this plotting window. It will be closed when you exit IRAF!***
- Hit *q* to quit *imexamine.*

The *imexamine* Task

•Imexamine provides multiple types of plots. You just have to type a key depending on what plot you want:

- *r* : radial profile plot.
- *l* : line plot.
- *c* : column plot.
- *s* : surface plot.
- *e* : contour plot.
- *j* : line 1d gaussian fit plot.
- *k* : column 1d gaussian fit plot.
- *h* : histogram plot.

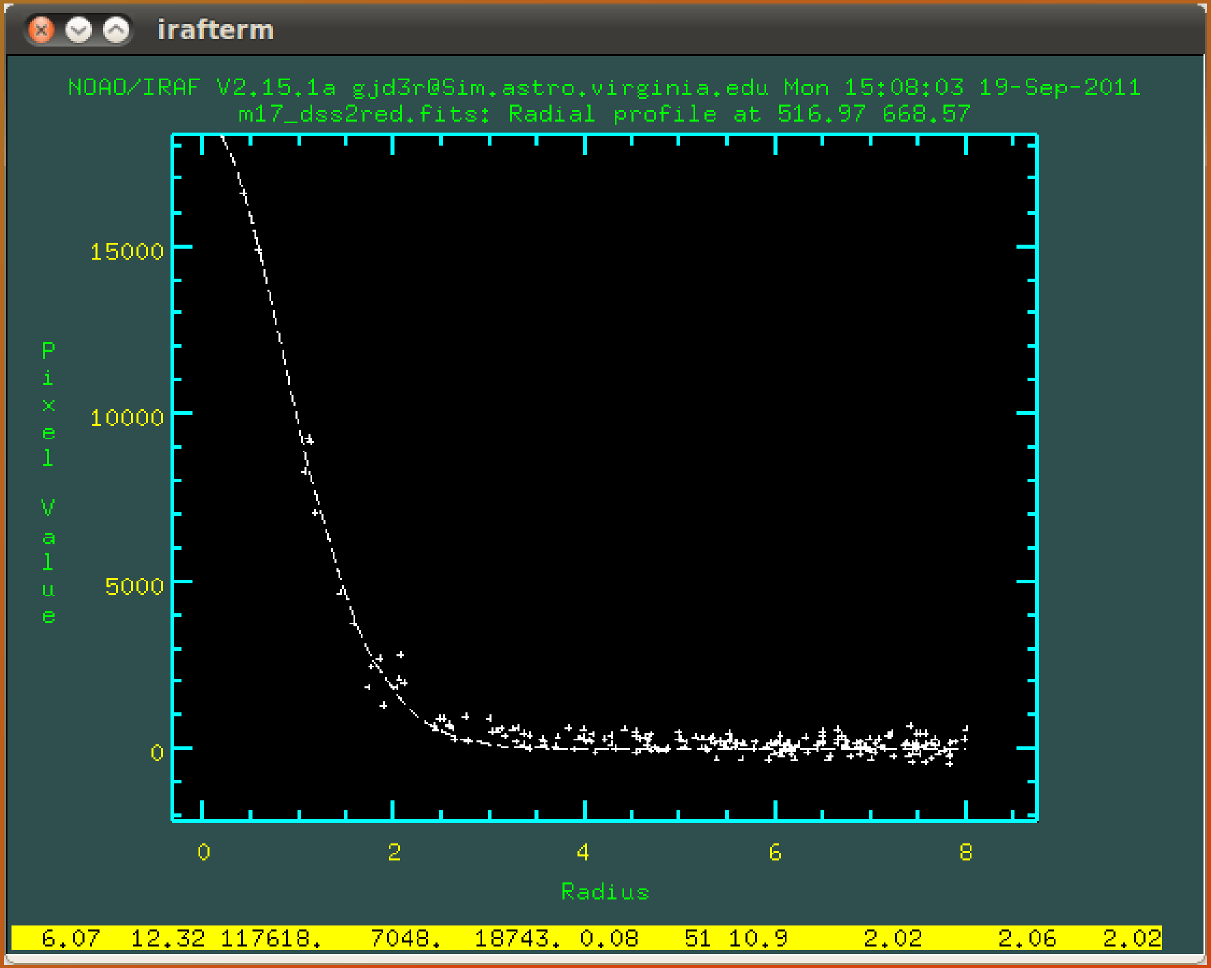
Figure on the right:

•A star radial profile plot.

•The last three numbers in yellow show the FWHM of the PSF in pixels.

•Such plots enable you to see if the image is in focus.

•Extremely important when you are at the telescope!



```
NOAO/IRAF V2.15.1a gjd3r@Sim.astro.virginia.edu Mon 15:08:03 19-Sep-2011
            m17_dss2red.fits: Radial profile at 516.97 668.57
```

```
 6.07  12.32 117618.   7048.  18743. 0.08   51 10.9     2.02     2.06   2.02
```

An *imexamine* radial profile plot.

The *imexamine* Task

- The previous plot commands have hidden parameter files, where you can customize the tasks.

These tasks are named by the letter for plotting, followed by *imexam*.

For example, to modify the parameters of the radial profile plot:

```
ecl> epar rimexam
```

- Some commands won't draw a plot, but print on the screen. For example:
  - *a* : prints *aperture photometry.*
  - *m* : prints *statistics* of the region around the cursor. It is very useful for characterizing your images for the data reduction.
- There are *colon-commands.* These commands are composed of the colon key and a name, or colon key, name and value.
- Type *?* when *imexamine* is running to print the help about all the possible commands!

The *implot* Task

- *Implot* is a task that “plot lines and columns of images”.
- *Implot* works interactively.
- *Implot* contains commands and colon-commands.
- However, *implot* does not display an image as *imexamine* does.

- See the *implot* help by typing *?* when *implot* is running.
  - In addition, you can see help about the *graphics cursor* by typing *:.help* when implot is running.
- We will see an example in the next slide.

The *implot* Task

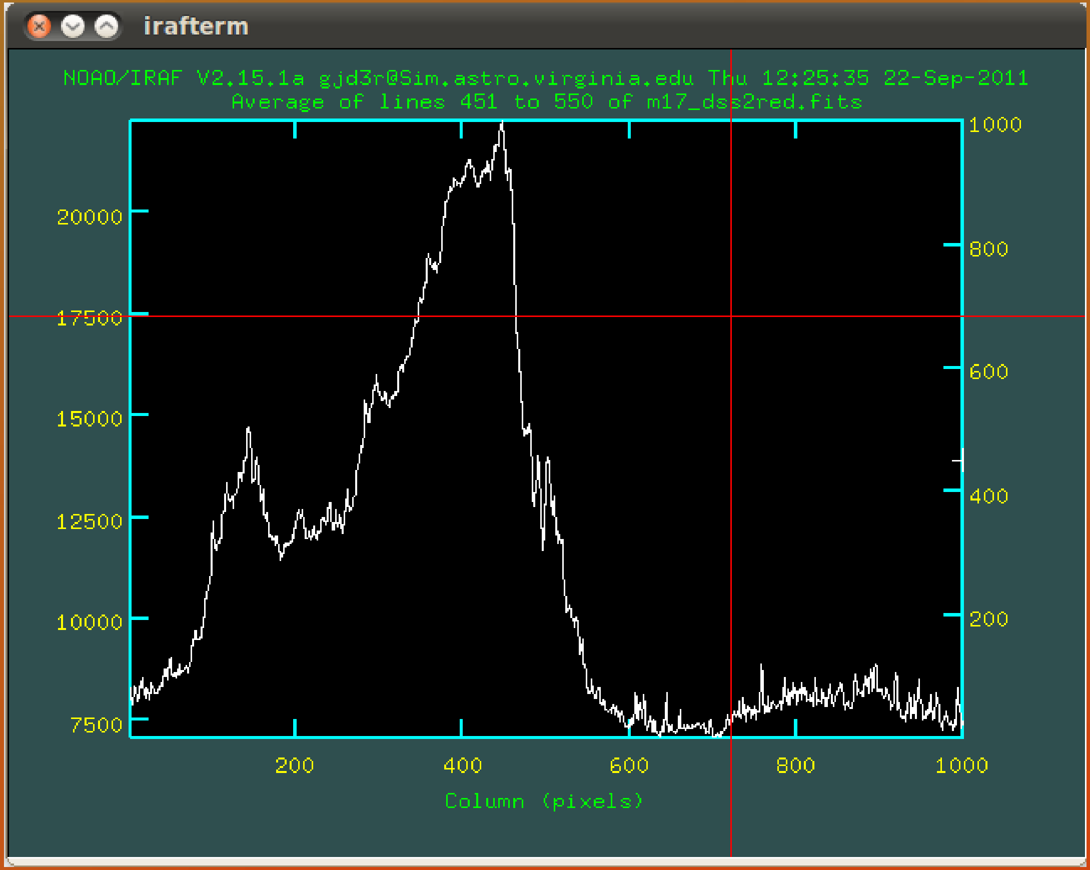
- As an example, we will plot the average over 100 lines centered at line 500 for our M17 red image.

  ecl> *implot m17_dss2red.fits*

- In the plotting window, type:
  - *:a 100*

    sets average to 100 lines or columns.
  - *:l 500*

    does a line plot centered at line 500.
  - We will see how to save the plot in the next slide.
  - Hit *q* to quit *implot.*



Line plot from *implot*

The axis on the right shows the line or column number for some of the implot interactive commands.

Saving a Plot As an .eps File

- You can save a plot into a .eps (*encapsulated PostScript)* with the command

  .snap eps

- For *imexamine* plots, you have to quit *imexamine* and type:

  ecl> *=gcur*

  - *gcur* enters the “interactive graphics cursor mode”.
  - Type *:.help* to display the whole help page.
  - Now, move to the plotting window and type *.snap eps*

- For *implot* plots, you don't have to quit *implot.* Simply type *.snap eps*
- A new file called *sgi#####.eps* (##### means a 5-digit number) will be created in your current working directory or in your */tmp* directory.
- If you need a .png or other image format, you can use the Linux task *convert:*

  % convert sgi#####.eps my_plot.png   (or .jpg, etc.)

Manipulating Image Files

- IRAF includes some special tasks to copy, rename and delete images.
- These tasks are *imcopy, imrename, imdelete.*
- In the past, image header unit and data unit were stored separated into two cross-linked files, where data units were kept in directories with large amounts of disk space.
- IRAF uses .imh extension for headers, and .pix for data.
- These specialized *im* tasks act on both .imh and .pix files together.
- Although IRAF now works with single FITS files, it is a good practice to use these tasks when you manipulate image files, and specify the .fits extension.

Image Arithmetic

- IRAF can perform image math as array operations (recall from last practical) between images or with constants and functions acting on arrays.

  Some examples:

ecl> *imarith image1.fits - image2.fits results1.fits*

ecl> *imar image1.fits * 1.5 results2.fits*

- *Division of images with graceful handling of divide by 0:*

ecl> *imdivide image1.fits image2.fits results1.fits*

- Sum/average/median images: (Note how we use a .txt file list as input)

ecl> *imsum @input_images.txt results3.fits option='median'*

- Apply a function to images: (See imfunction help for the definition of functions)

ecl> *imfunction image1.fits results4.fits log10*

- Apply boxcar smoothing to an image: (xwindow ywindow → 5 5)

ecl> *boxcar image1.fits results5.fits 5 5*

Other ecl Capabilities

- The *ecl* can perform mathematical operations.

Start the line with “=”

- Common functions are allowed: log, *log10, exp, sqrt, etc.*
- Trigonometric functions in radians: *sin, cos, tan, asin, acos, atan, etc.*
- Trigonometric functions in degrees: *dsin, dcos, dtan, dasin, dacos, etc.*
- *Beware* of *integer* versus *float* operatorions.

```
ecl> = 2/5
ecl> 0
ecl> = 2./5
ecl> 0.4
```

- Sexagesimal to decimal degrees conversion:

```
ecl> = 10:36:54
ecl> 10.615
```

Some Useful Packages

- *imred* : packages for data reduction of most of NOAO facilities data.
- *ccdred* : tasks to reduce data.
- *plot : general plotting tasks for image data.*
- *images* : packages for image coordinates, filtering, geometric transformation, matching, display utilities and utilities.
- *tables* : packages from the STSci for tabular data.
- *utilities* : a variety of tools. For example, data fitting routines.
- *astutil* : astronomical utilities. For example, airmasses, (l,b) to (ra,dec) convertion, etc.
- *twodspec* : two-d spectra packages. Aperture Extraction and Long Slit.
- *onedspec : one-d spectra tasks.*

Quitting IRAF

- Use the command *logout* to end your IRAF session.

  ecl> *logout*

  - Or:

  ecl> *lo*

- This will also close the plotting window if IRAF opened it before.
  - Remember, NEVER close this window by yourself!

References

- Image Reduction and Analysis Facility (IRAF) NOAO website:

  http://iraf.noao.edu/
- IRAF.net:

  http://iraf.net/
- IRAF intro exercise by Jeannette Barnes in:

  http://iraf.noao.edu/tutorials/tutorials.html
- IRAF built-in help.