IDL Primer's course

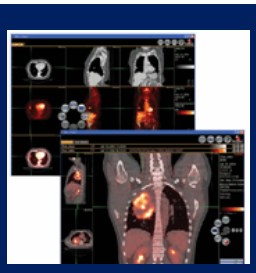
Luciano Nicastro
nicastro@iasfbo.inaf.it

Reference Web site:
http://ross.iasfbo.inaf.it/IDL/

IDL ?

IDL, the Interactive Data Language, is the ideal software for data analysis, visualization, and cross-platform application development. IDL combines all of the tools you need for any type of project - from "quick-look," interactive analysis and display to large-scale commercial programming projects. All in an easy-to-use, fully extensible environment.

IDL ?



Current version (October 2007): **7.0**

Web site: http://www.ittvis.com/idl/

OS: runs “almost” in the same way on UNIX, Windows and MAC platforms

Costs: ask ITT national representative; “network” and “personal” license are available. For educational Institutions, “campus” or “student” versions. Cost for profit companies are much higher!

Installation: via delivered DVD or download from the internet. Without license file, IDL can be used in demo mode for 7 minutes (cannot write files, etc.).

IDL ?

An interesting feature for developers is the IDL Virtual Machine:

It's a free tool which allows software developers to distribute compiled IDL code applets, or entire applications to colleagues and customers without additional licensing requirements or fees!

IDL official courses at various level are “offered” by ITT and last from 2 to 5 days. Costs are of the order of 1000 Euro. For information check the Web site (free webinar are available) and/or contact ITT Italia.

The main reference IS and MUST be the **"IDL Online Help“.**

No course can compare to practice, especially if combined with a well defined aim!

Manuals/Tutorials/Libraries

The IDL Online Help is a comprehensive resource. It runs in the web browser and is a major improvement introduced in version 6.x and makes its use really easy and more than "helpful". Just run `idlhelp` or type `?` at the `IDL>` prompt. The ITT page http://www.ittvis.com/idl/docs/index.asp has several free Guides. Printed copies can be bought.

As usual, the WEB is a valuable resource, even if (typically) the reference IDL version is not the latest. Try typing "IDL training" or "IDL tutorial" or similar in Google… You'll probably find all of the following:

Tutorials, On-line documentation, Books, User's libraries and programs, this course!

Tutorials

www.us-vo.org/summer-school/2006/proceedings/presentations

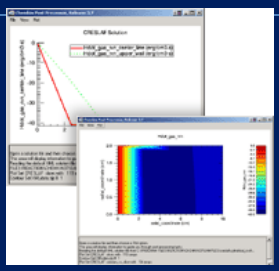
US-VO Tutorial (summer school 2006)

www.ncnr.nist.gov/staff/dimeo/IDL_Training.html

Rob Dimeo's training courses at NCNR (2 PDF files of the courses are available).

www.astro.virginia.edu/class/oconnell/astr511/IDLguide.html

R. W. O'Connell – A Guide to IDL for Astronomers



On-line documentation

- **General:**
  www.metvis.com.au/idl/
  IDL resources @ METVIS Services. A fairly complete list of web resources.
  www.dfanning.com/
  Coyote's Guide to IDL Programming. A growing list of suggestions and example progams (see below).

- **ITT IDL manuals in PDF format:**
  www.ittvis.com/idl/docs/index.asp
- **Direct Graphics in IDL:**
  www.sljus.lu.se/stm/IDL/Surf_Tips/
  Struan Gray's excellent tutorial on Extending IDL's Surface Plotting Routines.

On-line documentation (2)

- **Selected topics:**
  ftp://fermi.jhuapl.edu/s1r/idl/s1rlib/local_idl.html
  Tutorials on some of the JHU/APL/S1R IDL Library routines.

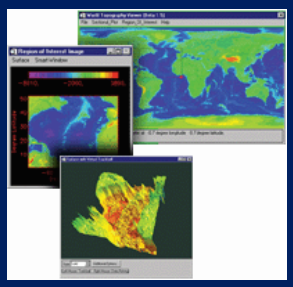
- Documentation and tips on IDL programming are also available at the **ITT site**:
  www.ittvis.com/codebank/index.asp
  and the **IDL Astronomy User's Library**:
  idlastro.gsfc.nasa.gov/homepage.html

- **Old IDL FAQ:**
  www.astro.virginia.edu/class/oconnell/astr511/IDLresources

On-line documentation (3)

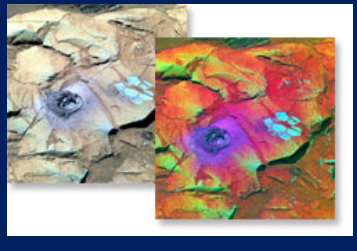
- **Old IDL newsgroup:**
  comp.lang.idl-pvwave

  Hosted by Google:
  groups.google.com/group/comp.lang.idl-pvwave/topics
  Mirrored at:
  www.fotovallescrivia.it/public/news/comp.lang.idl-pvwave/co



Books

- David Fanning: "IDL Programming Techniques". Available at: www.dfanning.com/documents/books.html (it was $65)
- Ronn L. Kling: "Application Development with IDL, Combining Analytical Methods with Widget Programming". Available at: www.rlkling.com/html/textbook.htm (it was $55)
- **ITT's training manuals:** see ITT web site.



User's libraries and programs

- IDL Astronomy User's Library:
  http://idlastro.gsfc.nasa.gov/homepage.html
  - A refernce site for all 'generic' Astronomical procedures
  - Procedures documentation
  - IDL-Databases of astronomical catalogues (small)
  - NEWS
- Eric Deutsch IDL Libraries Browser:
  www.astro.washington.edu/deutsch/idl/htmlhelp/index.html
  www.astro.washington.edu/deutsch/idl/htmlhelp/slibrary28.h

  ESRG UCSB Library
- Craig B. Markwardt IDL Library:
  http://cow.physics.wisc.edu/~craigm/idl/

User's libraries and programs (2)

- Soho library
  http://sohowww.nascom.nasa.gov/solarsoft/gen/idl/

  Various libraries, including WCS (World Coordinate System)
  www.mps.mpg.de/projects/soho/sumer/text/cookbook.html
  SUMER Data Cookbook
- IDL Libraries at IAAT, Astronomy:
  http://astro.uni-tuebingen.de/software/idl/
- Extended IDL Help:
  http://astro.berkeley.edu/~marc/idlshare/general/html/
- *Use a web search engine for more…*

Please, again note that sites may contain material not up-to-date in many respects. Do not contact me if something is missing or does not behaves as announced in the web sites or libraries or programs.

Linux specific notes

- **Idl_setup file:**

  it can be found in the directory: /usr/local/itt/idl/bin "**idl_setup**" or "**idl_setup.csh**" for the csh/tcsh shell and "**idl_setup.bash**" for bash. Even if on some system you already have the IDL commands defined, I suggest to copy this file into your $HOME dir. and edit it for customization. To execute it automatically for every terminal:

  csh/tcsh: at the end of the "`.cshrc`" (or `.login`) file add

  ```
  source ~/idl_setup.csh
  ```

  bash: in the "`.bashrc`" (or `.bash_profile`) file add

  ```
  . ~/idl_setup.bash
  ```

The setup file (tcsh)

```
# Change and uncomment below for a network license
#setenv LM_LICENSE_FILE 1700@ServerName
setenv ITT_DIR /usr/local/itt
setenv IDL_DIR /usr/local/itt/idl
alias ittlicense $IDL_DIR/bin/ittlicense
alias idl $IDL_DIR/bin/idl
alias idlde $IDL_DIR/bin/idlde
alias idlhelp $IDL_DIR/bin/idlhelp
alias idlman $IDL_DIR/bin/idlman
alias idlrpc $IDL_DIR/bin/idlrpc
alias idldemo $IDL_DIR/bin/idldemo
if (-d $HOME/IDL) then
  setenv IDL_PATH "<IDL_DEFAULT>":\+$HOME/IDL
endif
if (-e $HOME/.idl_startup) then
  setenv IDL_STARTUP $HOME/.idl_startup
```

Linux specific notes (2)

There are also X11 resources which are used by IDL (in the file **~/.Xdefaults** or in the file **$XAPPLRESDIR/Idl**).

idl.colors: number of color IDL can use (useful for PseudoColor devices).
idl.gr_depth: Depth, in bits, of the graphics device in use.
idl.retain: default parameter retain (Backing Store selection → graphics covered by other windows):

0= none, 1= by server, 2= by IDL.
idl.gr_visual: type of visual device to use:

StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor.
And so on. Check the help pages!

For example:

…

Linux specific notes (3)

…

```
Idl*fontlist: screen16
Idl.colors: -16
Idl.retain: 2
Idl.gr_visual: PseudoColor
```

IDL saves preferences into the file

$HOME/.idl/itt/pref-10-idl_7_0-unix/idl.pref

The same (and more) parameters can be defined using a 'startup' file using IDL instructions. For example:

…

The startup file

```
…
  print, 'Setting display attributes...'
; 24 bit true color display with backing store
  device, deco=0, retain=2, true=24
; Create window to allocate colors
  window, /free, /pixmap
; Might not be needed, but won't hurt
  plot, [0]
; Delete the window
  wdelete, !d.window
; Set the vector font size
  device, set_character_size=[6,9]
  print, 'Number of colors allocated is ', !
  d.n_colors
```

Linux specific notes (4)

Keyword DECOMPOSED=1 means IDL must interpret the color indices as composed by 3 values (8-bits each) corresponding to the red, green and blue intensities (from the less → most significative byte). Default for a display using TrueColor and DirectColor graphics (for example Windows and any machine equipped with graphics card with 16 million colors).

DECOMPOSED=0 means that IDL must interpret the color index (actually the less significant 8-bits) as the index of the PseudoColor color table → this index is in the range 0 – 255. This "was" the default for UNIX machines (not Linux!) using a "standard" display and allowed users (like me) to write "standard" code which run on any machine. Nowadays, TrueColor displays are the standard, so what runs on Linux should run on Windows in the same way (graphically speaking). The keyword PSEUDO is not allowed on Windows.

Linux specific notes (6)

If you prefer, can use idlde, which launches a GUI 'Workbench'. In your home directory you'll have a subdir. .idl/itt/idlworkbench-config-idl70 containing more directories with the application preferences etc.

Preferences can be changed either manually or using the Workbench itself.

IDL system variables

- Constants
  !PI, !DPI, !RADEG, !DTOR, !MAP, !VALUES
- For the graphics
  !D, !X, !Y, !Z, !P, !ORDER
- Error Handling and Informative messages
  !ERR, !ERR_STRING, !SYSERR_STRING, ...
- IDL environment
  !PATH, !PROMPT, !VERSION, ...

To add system variables:

```
IDL> DEFSYSV, 'Var_Name', Value [, /Read_only] [, EXISTS=i]
IDL> DEFSYSV, '!TEXTOUT', 1
```

Keep note...

1. IDL is not case sensitive (but, under Linux, the file names which store the IDL routines **are** case sensitive!)
2. IDL "procedures" are of type PROgram and FUNCTION (similarly to Fortran SUBROUTINE and FUNCTION) and both can have "comma separated" parameters and keywords. Parameters can be passed by reference or value, keywords can be passed by name or value using the format: KEY_NAME=name (value) or simply /KEY_NAME which translates into KEY_NAME=1 (i.e. TRUE). Functions have their parameters and keywords passed in brackets. For example:

PRO:

```
PLOT, FINDGEN(20)^2, XSTYLE=1, YSTYLE=1, THICK=2
```

FUNCTION:

```
cv = CONVERT_COORD( [0,1],[0,1],/NORM,/TO_DEV )
```

Keep note...(2)

A series of commands make a "script" which IDL calls a MAIN. Script syntax is slightly (but significantly) different from that used in procedures.

3. Arrays are defined by comma separated values in square brackets:
   1-d: a = [3,6,12,24,64], **2-d:** b = [[2,4,6,8,10],[8,12,24,36,72]] and the indices of the elements go from **0** to **n_elements - 1**. To extract an adjacent section one can use "**:**". For example to transfer into the variable c the first 3 elements of a:
   c = a[0:2] and to transfer those from the third to the last:
   c = a[2:*]. To extract random elements one can use an array of integer values (chosen elements). For example:
   i = [0,2,4] & c = a[i] will select first, third and fifth element of the array a. In a two dimensional array, the first index refers to the column, the second to the row (in Fortran it's the reverse).

Keep note...(3)

4. Constant numbers containing the "." or "E" (es. 13., 2e3) are assumed to be of type FLOAT (4 bytes); those containing a "D" of type DOUBLE (8 bytes); those with an "L" at the end of type LONG INTEGER (4 bytes) o simply LONG; those ending with a "B" are of type BYTE; those with none of these letters are of type INTEGER (2 bytes).
   Moreover: UL →UNSIGNED LONG, LL → 64-bit LONG, ULL → UNSIGNED 64-bit LONG.
   To define an exadecimal value, add an "X" at the end of the string constant. Example: '2E'XB, 'FF'XL, ecc.
   Similarly use "O" for octal constants. Please refer to the on-line help → **Constants**.

Pay attention to the definition of **INTEGER** constants which could trespass the **32768** limit (especially in the FOR loops!).

Keep note...(4)

Example:

```
a = 7 & a = a+32760 & print, a
32767
a = 7 & a = a+32761 & print, a
–32768
```

It would have been OK if I used "a = 7L"

If not sure, always add "L" at the and of integer values.

5. Array elements are addressed between square brackets [ ] (it was "()"). Ex.: a[10]

Pay attention to the definition of **INTEGER** constants which could trespass the **32768** limit (especially in the FOR loops!).

Keep note...(5)

6. Keyword names can be truncated to the letter which makes it "unique".
   For example if a command or procedure has as possible keywords CHARTHICK and CHARSIZE, then those can be truncated to CHART and CHARS, respectively.
7. Blank spaces (one or more) in expressions and statements containing operator, "=", ",", "&", etc. are optional;
   an instruction line can be split in several lines adding a "$" at the end of each line (not the last one, of course). Example:

```
PLOT, FINDGEN(20)^2, XSTYLE=1, YSTYLE=1, THICK=2, $
 XRANGE=[-2,22], TITLE='Demo plot'
```

8. Several instruction lines can be concatenated on the same line using an "&" as separator. Example:

```
a = fltarr(12) & b = dblarr(4,7)
```

Keep note...(6)

9. Comment lines (or part of it) start with a semicolon ";". Example:

```
plot, alog(x),alog(y), xtype=1, ytype=1; logarithmic plot X vs. Y
```

10. The keyword FORMAT, used to have formatted I/O, uses the same notation as Fortran OR C. Example:

```
PRINT, [1,2,3,4], FORMAT='(4(I2,2X))'
```

11. Alphanumeric strings are concatenated using a "+" sign. Example:

```
a = "Example" & b = 'of strings' & c = 'concatenation.'
ss = a +' '+ b +' '+ c & print, ss
Example of strings concatenation.
```

Keep note...(7)

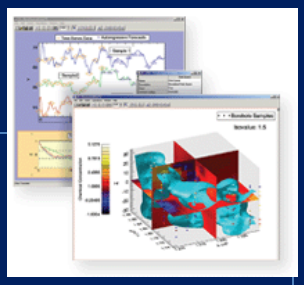
12. Plotting coordinates can be given using the following types:
    DATA (defined by the data values)
    DEVICE (defined as pixel coordinates of the active device)
    NORMAL (defined in the fixed interval [0, 1] for all the axes)

Examples:

```
plot, [0,10], [0,10], /data          ; default
plots, [.5,1], [.5,1], /norm         ; normalized
plots, [100,200], [100,200], /dev    ; device
```

13. For the on-line help "idlhelp" or, at the `IDL>` prompt:
    ? <Enter> or
    ? item <Enter>

iTools ?



The iTools are made up of five pre-built tools:

- iPlot provides 2D and 3D graphing
- iSurface allows for surface representations of 2D array data and irregularly sampled point collections
- iContour enables the production and manipulation of contour plots
- iImage provides image display, exploration, ROI definition and basic processing
- iVolume is used for the rendering and dissection of volumetric data
- iMap allows you to work with geo-located data
- iVector displays 2D vector flow fields

Let's start !

From now on, IDL **intrinsic** commands / instructions / PROgrams / FUNCTIONS will be reported in **CAPITAL LETTERS** whereas **external** (personal, etc.) programs will be reported in **lower-case letters**. Variable names will be lower-case too.

Run IDL interactively typing: `idl` or `idlde` (def. on Windows).

This results into the "terminal" (old-style) approach and the new "Workbench".

Operative System check

```
IDL> HELP, !VERSION, /STRUCT

** Structure !VERSION, 8 tags, length=104, data
   length=100:
      ARCH            STRING    'x86_64'
      OS              STRING    'linux'
      OS_FAMILY       STRING    'unix'
      OS_NAME         STRING    'linux'
      RELEASE         STRING    '7.0'
      BUILD_DATE      STRING    'Oct 25 2007'
      MEMORY_BITS     INT              64
      FILE_OFFSET_BITS
                      INT              64
```

Operative System check (2)

Within a program can check OS like this:

```
CASE !VERSION.OS_FAMILY OF
   'MacOS':...

   'unix':...

   'Windows':...
 ELSE:
 ENDCASE
```

Useful to set initial parameters which are OS dependent (directory names, display color properties, etc.).

Graphics device check

```
IDL> HELP, /DEVICE
Available Graphics Devices:
  CGM HP LJ NULL PCL PRINTER PS REGIS TEK X Z
  Current graphics device: X
  Server: X11.0, The XFree86 Project, Inc, Release 40300000
  Display Depth, Size: 24 bits, (1400,1050)
  Visual Class: TrueColor (4)
  Bits Per RGB: 8 (8/8/8)
  Physical Color Map Entries (Emulated / Actual): 256 / 256
  Colormap: Private, 16777216 colors.  Translation table: Enabled
  Graphics pixels: Decomposed,        Dither Method: Ordered
  Write Mask: 16777215 (decimal) ffffff (hex)
  Graphics Function: 3 (copy)
  Current Font: <default>,    Current TrueType Font: <default>
  Default Backing Store: Req from Server.
  Window Status: ---------------------
  id  typ(    x,   y,   backing store)
   0: Win( 700, 525, Req from Server)
```

Operators

```
IDL> a = INDGEN(10) ; array of integers 0-9
IDL> PRINT, a
   0    1    2    3    4    5    6    7    8    9
IDL> PRINT, 10 - a
   10   9    8    7    6    5    4    3    2    1
IDL> b = a < (10-a) ; the operator "<"
IDL> PRINT, b       ; let's see the result...
   0    1    2    3    4    5    4    3    2    1
IDL> v = 12
IDL> v = v > 1 < 10 ; operators "<" and ">"
IDL> PRINT, v       ; the result ...
      10
IDL> b = a > (10-a) ; the operator ">"
IDL> PRINT, b       ; the result...
   10   9    8    7    6    5    6    7    8    9
```

Operators (2)

Logical operators are like in Fortran (but without "."):

```
  LE GE GT LT NOT AND OR NOR
IDL> i = WHERE(b GT 7)
The WHERE function returns the indices of its array
  elements which satisfy the expression argument.
IDL> HELP, i
IDL> PRINT, i
         0         1         2         8         9
The indices of "b" satisfying the expression
Let's substitute these elements with the average of
  the values >= 7 (which is 5)
IDL> b[i] = TOTAL(WHERE(b le 7, n))/n
TOTAL performs the sum of its argument and the "n"
  parameter in the WHERE returns the number of
  elements in "b" which are >= 7
```

Operators (3)

```
IDL> PRINT, b
    5     5     5     7     6     5     6     7     5     5
```

Multi-dim arrays multiplication:

```
IDL> array1 = [ [1, 2, 1], [2, -1, 2] ]
Array with 3 columns and 2 rows
IDL> array2 = [ [1, 3], [0, 1], [1, 1] ]
Array with 2 columns and 3 rows
IDL> PRINT, array1#array2
           7          -1           7
           2          -1           2
           3           1           3
Matrix multiplication (cols. x rows)
```

Operators (4)

```
IDL> a = array1##array2
Matrix multiplication (rows x cols.)
IDL> HELP, a
A               LONG      = Array[2, 2]
```

Array expansion:

```
IDL> a = [ a,[[8,9],[10,11]] ]
Expand the array adding 2 extra columns
IDL> HELP, a
A               LONG      = Array[4, 2]
IDL> PRINT, a
           2           6           8           9
           4           7          10          11
```

Operators (5)

Selecting columns or rows:

```
IDL> PRINT, a[*,0]  ; first row
           2           6           8           9
IDL> PRINT, a[1,*]  ; second column
           6
           7
```

Dimension degeneracy:

To remove an initial "single-element" dimension (referred as "degeneracy") the intrinsic REFORM function can be used:

Operators (6)

```
IDL> ar1 = a[1,*]
Assign the second column of "a" to "ar1"
IDL> HELP, ar1
Note how we get a 2-d array having 1 column
AR1             LONG      = Array[1, 2]
IDL> ar2 = REFORM(ar1)
IDL> HELP, ar2
AR2             LONG      = Array[2]
```

Edit a script and run it

```
IDL> $vi example.pro
Cut-and-paste one of the on-line help
IDL> @example
```

The $ allows to “escape” the shell, i.e. to execute a command in a new shell. The @ tells IDL to execute the script example.

Help on programs

To view the help of the routines for which the code exists on the IDL path (!PATH), one can use DOC_LIBRARY. For example to see the help on DOC_LIBRARY itself:

```
IDL> DOC_LIBRARY, 'doc_library'
--- Documentation for /usr/local/itt/idl/lib/doc_library.pro ---

 NAME:
       DOC_LIBRARY

 PURPOSE:
    Extract the documentation template of one or more IDL modules
    (procedures or functions).  This command provides a standard
   interface
    to the operating-system specific DL_DOS, DL_UNIX, and
    DL_VMS procedures.
...
```

Help on programs (2)

Note that the extension ".pro" must be omitted (and the final ' can be omitted). Moreover if the file contains more routines with their header help section (we'll see later that the header comment starts conventionally with ;+ and end with ;-), ONLY the first one is shown!

Again, under Linux it is important to pay attention to UPPER and lower case letters. Using always lower case letters in file names is advisable.

Again, for the system routines, the best way to get help is using the interactive web interface:

From the Linux prompt: idlhelp

From the IDL prompt: ?

Help on programs (3)

Examples of help on procedures that can be found in the IDL_PATH / !PATH system variable (Linux / IDL):

```
DOC_LIBRARY, 'nxyreadf

DOC_LIBRARY, 'mpltw'

DOC_LIBRARY, 'pposn'

DOC_LIBRARY, 'sqplset'

DOC_LIBRARY, 'aitoff_grid2'

DOC_LIBRARY, 'lmfit' ; IDL routine with source code

...
```

And there is much more…

MIN / MAX and Statistics (assume data into array d):

```
print, MIN( d, MAX=hmax ), hmax
hmin = MIN( d, MAX=hmax )
moms =  MOMENT( d, MDEV=mdev, SDEV=sdev )
  ninfo = 7  &  info = STRARR(ninfo)
  info[0] = 'Mean: '+ STRTRIM(STRING(moms[0]),2)
  info[1] = 'Variance: '+ STRTRIM(STRING(moms[1]),2)
  info[2] = 'Skewness: '+ STRTRIM(STRING(moms[2]),2)
  info[3] = 'Kurtosis: '+ STRTRIM(STRING(moms[3]),2)
  info[4] = 'Mean Absolute Deviation: '+
            STRTRIM(STRING(mdev),2)
  info[5] = 'Standard Deviation: '+
            STRTRIM(STRING(sdev),2)
  info[6] = 'Minimum: ' + STRTRIM(hmin,2) +
            'Maximum: ' + STRTRIM(hmax,2)
```

And there is much more…(2)

- Variables information (size , type, …):
  `SIZE`
- Memory (allocation) saving:
  `TEMPORARY`
- Strings manipulation:
  `STRTRIM, STRCOMPRESS, STRMID, STRPOS, STRPUT, STRSPLIT`
- Interacting with the operative system:
  `SPAWN`
- Logging commands into a file:
  `JOURNAL`
- Selecting files for I/O:
  `FILEPATH, FINDFILE, PICKFILE`