Basic IDL Commands

SIMG 726

Objective

• Quick Tour of IDL
• Writing IDL Programs

Starting IDL in UNIX

• To start command line interface IDL environment
  % idl
• To start graphical interface IDL environment
  % idlde

IDL Help

• MOST IMPORTANT IDL COMMAND OF ALL

  IDL> ?

IDL Help Window

Other IDL Help Facilities

IDL> doc_library,’’moment’’

• Allows specially formatted code comments to be viewed without having to view the source code
General IDL Information

- Platform specific limitation
  - UNIX
  - File name conflict between lower and upper case characters
  - e.g. addnum.pro vs. AddNum.pro
  - Recommend use of strictly lowercase file names

IDL Character Set

- IDL is not case sensitive
- Commands and arguments are delimited by commas.
  - e.g. IDL> print, a
- Comments are started by a semicolon (;)
  - e.g. IDL> ; This is a comment
- Command options are set using a backslash (/)
  - e.g. IDL> tv, a, /order

Other IDL Characters

- System variables are started with an exclamation point (!)
  - e.g. IDL> print, !PI
  - e.g. IDL> y = sin( x!*DTOR )
- Command line continuation with $
  - e.g. IDL> print, 180 / $ !PI

Other IDL Characters

- Multiple IDL commands can be separated by the & symbol
  - e.g. IDL> a=0.0 & b=1.0
- UNIX Command execution
  - e.g. IDL> $ls -la

Journal Facility of IDL

- The journal command allows you to record all IDL activity during a particular session
  - IDL> journal, 'sequence.pro'
  - IDL> print, 3*5
  - IDL> journal

sequence.pro

; IDL Version 3.6.1a (MacOS Macintosh)
; Journal File for Rolando Raqueño@green
; Working Directory: Raqueño's Hard Drive:Applications
; Date: Tue Sep 17 00:56:18 1996
print,3*5
; 15
Sample Commands

- Some sample command
  IDL> print, 3*5
  IDL> print, SystemTime()
  IDL> num = 5*12
  IDL> help, num

Creating an Array

- Some sample command
  IDL> num = fltarr(40)
  IDL> help, num

Shortcut Creating Indexed Array

- Some sample command
  IDL> num = findgen(40)
  IDL> help, num

Indexed Array Example

- Some sample command
  IDL> num = findgen(40)*10
  IDL> print, num

Creating an “Indexed Array”

- Some sample command
  IDL> For i=0, 39 do num(i)=i
  IDL> help, num

Indexed Array Example

- Some sample command
  IDL> num = findgen(40)*10-195
  IDL> print, num

• In general you will be using a linear form:
  \( y = mx + b \)
**Extracting Array Subset**

- Some sample command

```idl```
IDL> num1 = num(0:36)
IDL> print, num1
-195.000     -185.000     -175.000     -165.000     -155.000     -145.000     -135.000
-125.000     -115.000     -105.000     -95.0000     -85.0000     -75.0000     -65.0000
-55.0000     -45.0000     -35.0000     -25.0000     -15.0000     -5.00000      5.00000
15.0000      25.0000      35.0000      45.0000      55.0000      65.0000      75.0000
85.0000      95.0000      105.000      115.000      125.000      135.000      145.000
155.000      165.000
```idl```

**Scalar Operation on Arrays**

- Some sample commands

```idl```
IDL> num = findgen(37)*10
IDL> print, num
0.00000      10.0000      20.0000      30.0000      40.0000      50.0000      60.0000
70.0000      80.0000      90.0000      100.000      110.000      120.000      130.000
140.000      150.000      160.000      170.000      180.000      190.000      200.000
210.000      220.000      230.000      240.000      250.000      260.000      270.000
280.000      290.000      300.000      310.000      320.000      330.000      340.000
350.000      360.000
```idl```

**Two-Dimensional Plotting**

- Displaying Two-Dimensional Graphics

```idl```
IDL> line = Sin(num * !DtoR)
IDL> plot, line
```idl```

**Two-Dimensional Plotting Options**

```idl```
IDL> plot,num,line,/Psym
```idl```

```idl```
IDL> plot,num,line,Psym=1
```idl```
Other Plotting Options

\[ IDL > \text{plot}, \text{num}, \text{line}, \text{linestyle}=1 \]

Labeling IDL Plots

\[ IDL > \text{plot}, \text{num}, \text{line}, \text{xrange}=[0,360], \] 
\[ \text{ytitle}='\text{Sin(x)}', \text{Title}='\text{Sine Plot}' \]

Other Plot Facilities

\[ IDL > \text{plot}, \text{num}, \text{line}, \text{xrange}=[0,360], \] 
\[ ytitle='\text{Sin(x)}', \text{Title}='\text{Sine Plot}' \] 
\[ IDL > \text{plots}, [0,360], [0,0] \]

Polygon Filling

\[ IDL > \text{loadct, 5} \] 
\[ IDL > \text{polyfill}, \text{num}(0:18), \text{line}(0:18), \text{color}=120 \] 
\[ IDL > \text{polyfill}, \text{num}(18:36), \text{line}(18:36), \text{color}=50 \]

Data File Inputs

- Copy the following files from 
  ~rvrpci/pub/IDL/training
  surface.dat
  ctscan.dat

Three-Dimensional Plotting

- Displaying Three-Dimensional Graphics
  - To get the data into IDL, you will need to open a file and read it into an array using the following commands (N.B. this is specific for this particular data set)

\[ IDL > \text{file}='\text{surface.dat}' \] 
\[ IDL > \text{open}, \text{lun}, \text{file}, /\text{get_lun} \] 
\[ IDL > \text{peak}=\text{fltarr}(40,40) \] 
\[ IDL > \text{readf}, \text{lun}, \text{peak} \] 
\[ IDL > \text{free_lun}, \text{lun} \]
Surface Plots
• Displaying Three-Dimensional Graphics

IDL> print, max(peak), min(peak)
IDL> surface, peak

A Quick Tour of IDL
The following surface command lets you manipulate the background and foreground colors

IDL> surface, peak, background=200, color=45, title='Surface Projection', charsize=2.0

Shaded Surface Plotting

IDL> shade_surf, peak

Draping Data Over a Surface
We can drape a given surface over another one by the following command (we read in the data snow the same manner as peak)

IDL> shade_surf, peak, $shade=snow
Surface Plot Colored by Altitude

– We can generate a simple elevation color map.

IDL> shade_surf, peak, shade=bytscl(peak)

Contour Plots

– We can generate a simple contour map.

IDL> contour, peak

Contour Plots

– We can generate a simple contour map and adjust spacing.

IDL> contour, peak, nlevels=10

Contour Plots

– We can generate a simple contour map and adjust spacing along with line styles.

IDL> contour, peak, nlevels=10, /follow

Colored Contour Plots

– We can generate a simple contour map and adjust spacing along with line color.

IDL> contour, peak, nlevels=10, c_linestyle = indgen(5)

Colored Contour Plots

– We can generate a simple contour map and adjust spacing along with line color.

IDL> contour, peak, nlevels=10, c_colors = indgen(5)*20+80
Images Data

• Displaying Image Data
  – You can read in image data in a manner similar to other forms of two-dimensional data. You will need to ascertain whether the data is an ASCII file or a binary file

    IDL> open, lun, 'ctscan.dat', /get_lun
    IDL> scan = bytearr(256,256)
    IDL> readu, lun, scan
    IDL> free_lun, lun


Image Data

• Displaying Image Data

    IDL> window, /free, xsize=256, ysize=256
    IDL> tvscl, scan

    IDL> tvscl, hist_equal(scan)


Image Data

• Displaying Image Data

    IDL> window, /free, xsize=256, ysize=256
    IDL> tvscl, scan
    IDL> tvscl, sobel(scan)


Image Processing

• Displaying Image Data

    IDL> window, /free, xsize=256, ysize=256
    IDL> tvscl, scan
    IDL> tvscl, sobel(scan)


XLOADCT Routine

• Displaying Image Data

    IDL> xloadct
    IDL> tvscl, scan


Writing IDL Programs
General Form of an IDL Program

- Parts of an IDL program
  pro program_name [ , arg1, arg2, ..., argn]
  IDL statement #1
  IDL statement #2
  ...
  IDL statement #n
  end

General Form of an IDL Function

- Parts of an IDL function
  function program_name [ , arg1, arg2, ..., argn]
  IDL statement #1
  IDL statement #2
  ...
  IDL statement #n
  return, answer
  end

Compiling an IDL Program

- Create a program to take celsius arguments and print out fahrenheit values
  pro celsius_to_fahrenheit, celsius
  print, 9/5*celsius+32
  end
- To compile:
  IDL> .compile celsius_to_fahrenheit.pro

Compiling an IDL Function

- Create a function to take celsius arguments and print out fahrenheit values
  function celsius_to_fahrenheit, celsius
  answer=9/5*celsius+32
  return, answer
  end
- To compile:
  IDL> .compile celsius_to_fahrenheit.pro

Executing an IDL Program

- To use the program in scalar mode:
  IDL> c = 100.0
  IDL> celsius_to_fahrenheit, c
- To use the program in array mode:
  IDL> c = findgen(201)/2
  IDL> celsius_to_fahrenheit, c

Executing an IDL Function

- To use the program:
  IDL> c = findgen(201)/2
  IDL> f = celsius_to_fahrenheit(c)
  IDL> print, f
  IDL> plot, c, f
- Equivalently:
  IDL> print, celsius_to_fahrenheit(c)
  IDL> plot, c, celsius_to_fahrenheit(c)
Program Design
program file vs. function file
• Use functions in case of mathematical functions to allow composition of functions
  \[ k = \text{celsius_to_kelvin}(\text{fahrenheit_to_celsius}(f)) \]
• Use program file in case of process not requiring the return of data
  \[ \text{write_to_file('temperature.dat', k)} \]

Parameter Passing in IDL
• IDL passes parameters into functions and programs by reference
• This means parameters can be modified through “side effect”
  \[
  \text{pro celsius_to_fahrenheit, celsius}
  \begin{align*}
  &\text{celsius} = 9/5\times\text{celsius} + 32 \\
  &\text{end}
  \end{align*}
  \]

A Debugging Exercise
• The IDL function and program examples `celsius_to_fahrenheit` all compile, but generate incorrect answers in some cases.
• This is left as a debugging exercise

Hints for Statistics Problem
• Study the IDL functions
  – n_elements()
  – size()
• Try to reuse as many of the functions as possible.
• Do not be concerned with efficiency at this point.