

Next: [Contents](#)

INTRODUCCION A I.D.L.

Esta introducción a IDL se puede bajar entero en formato [dvi](#)(50Kb) o [Postscript](#) (comprimido, 93Kb).

I have written a [procedure which helps set device to PS](#) without the need to edit the main program. I put this program public so that anybody can use it and others can improve it. I need your input for that. Please send your comments to andry@fisica.ulpgc.es

- [Contents](#)
- [List of Figures](#)
- [INTRODUCCION A I.D.L 1](#)
 - [Comandos Utiles:](#)
 - [Datos: tipo y estructura](#)
 - [Cálculos en IDL:](#)
 - [Operaciones simples:](#)
 - [Operaciones mas complejas:](#)
- [INTRODUCCION A I.D.L 2](#)
 - [Comandos:](#)
 - [Array, Vector y Matrices:](#)
 - [Subíndice:](#)
 - [Multiplicación matricial #:](#)
 - [Simple 2-D gráficas:](#)
 - [Grabar sesión en IDL:](#)
 - [Programa simple en IDL:](#)
- [INTRODUCCION A I.D.L 3](#)
 - [Comandos:](#)
 - [Opciones para comandos de gráficas:](#)
 - [Programación:](#)
 - [Inicialización:](#)

- [Procesamiento:](#)
 - [Gráficos:](#)
 - [Fin de programa:](#)
 - [Ejecución de un programa en IDL:](#)
 - [INTRODUCCION A I.D.L 4](#)
 - [Comandos:](#)
 - [Bloque grupo de comandos:](#)
 - [INTRODUCCION A I.D.L 5](#)
 - [Comandos](#)
 - [Caracteres usados para los programas:](#)
 - [Subrutina y Función:](#)
 - [INTRODUCCION A I.D.L 6](#)
 - [Comandos](#)
 - [Ejercicios para programación en IDL](#)
 - [Objetivo 1:*Hacer una gráfica donde se superponen diferentes líneas con colores diferentes al del borde de la gráfica.*](#)
 - [Objetivo 2:*Comprender la idea de subrutina/función con argumentos y opciones.*](#)
 - [INTRODUCCION A I.D.L 7](#)
 - [Objetivo n1: Gráfica de isolínea con datos repartidos en una malla irregular:](#)
 - [Objetivo n2: Creación de nuevo eje:](#)
 - [Objetivo n3: Rellenar isolíneas con color o banda:](#)
 - [Objetivo n4: Utilización del comando CURSOR:](#)
 - [INTRODUCCION A I.D.L 8](#)
 - [About this document ...](#)
-

1999-03-07

Next: [List of Figures](#) **Up:** [Introducción a I.D.L.](#) **Previous:** [Introducción a I.D.L.](#)

Contents

- [Contents](#)
- [List of Figures](#)
- [INTRODUCCION A I.D.L 1](#)
 - [Comandos Utiles:](#)
 - [Datos: tipo y estructura](#)
 - [Cálculos en IDL:](#)
 - [Operaciones simples:](#)
 - [Operaciones mas complejas:](#)
- [INTRODUCCION A I.D.L 2](#)
 - [Comandos:](#)
 - [Array, Vector y Matrices:](#)
 - [Subíndice:](#)
 - [Multiplicación matricial #:](#)
 - [Simple 2-D gráficas:](#)
 - [Grabar sesión en IDL:](#)
 - [Programa simple en IDL:](#)
- [INTRODUCCION A I.D.L 3](#)
 - [Comandos:](#)
 - [Opciones para comandos de gráficas:](#)
 - [Programación:](#)
 - [Inicialización:](#)
 - [Procesamiento:](#)
 - [Gráficos:](#)
 - [Fin de programa:](#)
 - [Ejecución de un programa en IDL:](#)
- [INTRODUCCION A I.D.L 4](#)
 - [Comandos:](#)

- [Bloque grupo de comandos:](#)
 - [INTRODUCCION A I.D.L 5](#)
 - [Comandos](#)
 - [Caracteres usados para los programas:](#)
 - [Subrutina y Función:](#)
 - [INTRODUCCION A I.D.L 6](#)
 - [Comandos](#)
 - [Ejercicios para programación en IDL](#)
 - [*Hacer una gráfica donde se superponen diferentes líneas con colores diferentes al del borde de la gráfica.*](#)
 - [*Comprender la idea de subrutina/función con argumentos y opciones.*](#)
 - [INTRODUCCION A I.D.L 7](#)
 - [Objetivo n1: Gráfica de isolínea con datos repartidos en una malla irregular:](#)
 - [Objetivo n2: Creación de nuevo eje:](#)
 - [Objetivo n3: Rellenar isolíneas con color o banda:](#)
 - [Objetivo n4: Utilización del comando CURSOR:](#)
 - [INTRODUCCION A I.D.L 8](#)
-

1999-03-07

```
PRO main2proc, fichero, tempo, cmd_cp, cmd_rm
; THIS PROCEDURE CHANGE A MAIN PROGRAM INTO A PROCEDURE.
;     IT IS USED FOR PRINTING.

ln=''

                                                    print,'PROG=',fichero

openr,u1,fichero+'.pro',/get_lun
rdnwln:
readf,u1,ln
ln0= STRLOWCASE(strmid(strtrim(ln,1),0,4))

; CHECK IF IT IS A MAIN PROGRAM OR A PROCEDURE.
IF ln0 EQ 'pro ' THEN BEGIN
; CHECK IF IT NEEDS SOME INPUT KEYWORDS.

    ln1=str_sep(strcompress(strtrim(ln,2)),' ')
    prockwd=str_sep(strcompress(ln1(1),/remove_all),',')
                                                    print,'prockwd=',prockwd

    help,n_elements(prockwd)
    nel=n_elements(prockwd)
    case nel of
    1: goto,chgfl
    2: begin
        if strpos(prockwd(1),'prnt=prnt') eq -1 then begin
; CASE WHERE THE KEYWORD IS NOT A prnt=prnt
            kwrđ=' '
            kwrđ=prockwd(1)
            optkwrđ=str_sep(kwrđ,'=')
            if n_elements(optkwrđ) eq 1 then begin
; THE KEYWORD IS A POSITIONAL PARAMETER, IT NEEDS TO BE INPUT
                param=' '
                read,param,prompt=prockwd(1)+'='
                addln=prockwd(1)+'='+string(param)
            endif
; IN THE OTHER CASE, IT IS AN OPTIONAL PARAMETER, THUS NO NEED TO BE
;     SPECIFIED.
                goto,chgfl
            endif else begin
                tempo=fichero
                goto,exec
            endelse
        end
    else: begin
; THE PROCEDURE HAS MORE THAN 1 KEYWORD.
        kwrđ=strarr(n_elements(prockwd)-1)
        kwrđ(*)=prockwd(1:*)
                                                    print,'kwrđ>',kwrđ

        eqpos=strpos(kwrđ,'=')
; CHECK WHICH KEYWORD(S) IS(ARE) OPTIONAL.
                                                    print,'eqpos=',eqpos

        which=where(eqpos lt 0)
                                                    print,'which=',which

        kwrđ00=''
        if which(0) ne -1 then begin
            nbwhch=n_elements(which)
            kwrđ0=strarr(nbwhch)
            addln=strarr(nbwhch)

            for i=0,nbwhch-1 do begin
                read,prompt=kwrđ(which(i))+'= ',kwrđ00
                kwrđ0(i)=kwrđ00
            end
        end
    end
end
```

```

                                print, 'kwrđ0=', kwrđ(i)
                                addln(i)=kwrđ(which(i))+ '=' +string(kwrđ0(i))
                                endfor
                                print, addln
endif
which=where(eqpos gt 0)
nbopt=n_elements(which)
                                print, 'which gt 0=', which
if which(0) ne -1 then begin
    kwrđ1=strarr(nbopt)
    addlopt=strarr(nbopt)
                                for i=0, nbopt-1 do begin
                                    read, prompt=kwrđ(which(i))+ '>', kwrđ00
                                    kwrđ1(i)=kwrđ00
addlopt(i)=strmid(kwrđ(which(i)), 0, eqpos(which(i)))+ '=' +kwrđ00
                                endfor
                                print, addlopt
endif
end
endcase
endif
IF ln0 ne 'end' then goto, rdnwln
free_lun, u1
chngfl:
openr, u1, fichero+'.pro', /get_lun
openw, u2, 'xxxxxxx.pro', /get_lun
tempo='xxxxxxx'
printf, u2, 'pro ', tempo+', prnt=prnt'
if n_elements(addln) ne 0 then for i=0, n_elements(addln)-1 do printf, u2, addln(i)
if n_elements(addlopt) ne 0 then for i=0, n_elements(addlopt)-1 do $
    if kwrđ1(i) ne '' then printf, u2, addlopt(i)
while not eof(u1) do begin
    readf, u1, ln
    if strlowercase(strmid(strtrim( ln, 1), 0, 4)) ne 'pro ' then printf, u2, ln
endwhile
free_lun, u1, u2
; FROM HERE YOU SHOULD BE SURE THAT YOUR PROGRAM HAS THE NECESSARY
; PROCEDURE LINE WHICH IS:          pro program_name, prnt=prnt
exec:
RETURN
END
PRO pr_print, w_device=w_device, landscape=landscape, prnt=prnt, sz=sz, $
    flnm= flnm, xxx_comp=xxx_comp, no_plot=no_plot, compile=compile, $
    out_flnm=out_flnm, color=color, xoffset=xoffset, yoffset=yoffset
;+
; NAME:
;     PR_PRINT
;
; PURPOSE:
;     By running this procedure to call another specified procedure (the
;     real program to run), you will be able to choose after seen the
;     plotting graphics whether you want to print it or not.
;
; CATEGORY:
```

```
;      program handling, output.
;
; CALLING SEQUENCE:
;      pr_print
;
; OPTIONAL INPUT PARAMETERS:
;      prnt:   The size scale. It is defined as:
;              x-size
;              prnt=  -----
;              y-size
;              Sometimes, this parameter is computed within the main
;              program (especially for a map plot).
;              If not set manually nor within the main program, a value
;              of
;                  PRNT=0.75
;              is used.
;              The value introduced at the execution of this PR_PRINT
;              program will override any computed value.
;
;      sz:     A 2-element vector which specifies the X- and Y-size of
;              the hardcopy graphic (!!!! in PS mode !!!)
;
;      fichero:      A short cut by specifying at execution time the name of
;              the file of the program to plot.
;
; KEYWORD:
;      landscape:   If set and not equal to 0, generates a graphic in
;              a landscape mode.
;
;      w_device:   The output device used to show the plot. If not
;              specified, the plot will be first shown on the screen.
;
;      xxx_comp:   set this keyword to 1 if you know that it is not
;              the 1st time you use this 'pr_print' program during the
;              session.
;
;      no_plot:    Set this to 1 if you do not want to plot the graphs before
;              printing it. This is useful when your program needs a lot of time
;              to run, you are already sure of your result, and you want to plot
;              it right away.
;
;      compile:    Set this to 1 if you do need to compile the procedure.
;              The compilation should be done whenever you change to a new
;              procedure (or have made a change to the procedure) and you have
;              already used "pr_print" during your session.
;
;      out_flnm:   Set this to the name of the output filename. If not set the
;              default is the name of the program to be run with the specific
;              extension. ; *****THALASSA
;
;      color:     Set this to 1 to save color postscript file.      ;;COLOR
;
;      xoffset:   Set this to the offset distance in X (in Cm). If not set the
;              default is drawing the graph in the center of the paper.
;
;      yoffset:   Set this to the offset distance in Y (in Cm). If not set the
;              default is drawing the graph in the center of the paper.
;
; RESTRICTIONS:
;      Main programs (not carrying the leading line:
;              pro fichero
;      can be run. Another copy of the program is made converted into
;      procedure. The later is run and deleted after execution.
;      You should have some free memory on your drive for the temporary
;      copy to be run.
;
;      The program to run should have the following first executable
```

```
; line:
;         pro fichero,prnt=prnt
; If not, the optional keyword ",prnt=prnt" is added to the
; "pro fichero"-line.
;
; Main programs which contain Procedure(s) and/or Fuction(s) at the
; begining of the main file are not supported.
;
; << WARNING !!!>>
; << WARNING !!!>> IF YOU WANT TO SPECIFY THE SIZE OF YOUR GRAPH,
; << WARNING !!!>> DON'T FORGET TO DEFINE prnt IN YOUR MAIN
; << WARNING !!!>> PROGRAM.
; << WARNING !!!>>
;
; RESULT:
; The program will plot the graphic on the current device.
;
; MODIFICATION:
; 990520, allow user to specify xoffset/yoffset.
; 981008, corrected the change of background color.
; 980520, by WAR, Add "color" keyword.
; 980505, add "out_flg" keyword.
; Feb. 16th, 1998, rename "landscp" keyword => "landscape".
; Jan. 9th, 1998, add "compile" keyword for IDL V 5.0
; Dec. 20th, 1997, Add "no_plot" keyword.
; Oct. 30th, 1997, Correct bug about "PRO..." filename.
; Sept. 23rd, 1995, created by WAR.
; Oct. 15th, !version.os introduced
; Jan 1st, 1996, w_device, landscp, prnt, and sz keywords introduced.
; Jan 7th, 1996, Main plotting prog (which does not carry a
;         leading line "pro fichero") can be run.
; Feb 29th, 1996, add 'xxx_comp' keyword so that one can re-compile
;         the new-to-plot program.
;
; -----
;
; THIS PROGRAM WORKS PRETTY FINE BUT NEEDS ANYBODY'S INPUT FOR IMPROVEMENT
; AND FOR EXAMPLE, FOR MAKING A widget INTERFACE FOR END USERS.
; IT STILL NEEDS SOME IMPROVEMENT SUCH AS ACCPETING ALL THE GRAPHIC
; KEYWORD FOR device AND call_procedure.
;
;-
; THIS PROGRAM WILL BE USED AS IS FOR THE TIME BEING

if keyword_set(prnt) then begin
    prnt0=prnt
    set_prnt=1      ; FORCE THE SIZE SCALE TO THE prnt GIVEN VALUE
endif

if not keyword_set(w_device) then w_device=''
if not keyword_set(landscape) then landscape=0

old_pant= !d.name          ;*****
case STRLOWCASE(STRMID( !version.os, 0, 3)) of          ;*****
'win': begin
    ;pant='win'          ;*****
    sep='\ '
    cmd_cp='copy'
    cmd_rm='del'
    end
else: begin
```



```
if not keyword_set(sz) then begin
  if keyword_set(landscape) and landscape eq 1 then begin
; THE MAX SIZE FOR A LANDSCAPE MODE IS: (X,Y) (23,18) cm.
    xsz=23.
    ysz=xsz/prnt < 18.
    xsz=ysz*prnt
  endif else begin
; THE MAX SIZE FOR A PORTRAIT MODE IS: (X,Y) (18,23) cm.
    ysz=23.
    xsz=ysz*prnt < 18.
    ysz=xsz/prnt
  endelse

; UNTAG THIS LINE IF YOU WANT TO SEE THE SIZE OF YOUR PRINTING POSTSCRIPT
; GRAPH.
; print,'prnt=',prnt,' xsz=',xsz,' ysz=',ysz
;-----END OF UNTAG-----

endif else begin
; IF THE LEYWORD sz IS DEFINED, USE THE VALUES TO SPECIFY THE PLOT SIZE.
  xsz=sz(0)
  ysz=sz(1)
endelse

; Set the output filename.
IF NOT KEYWORD_SET(out_flg) THEN out_flg= fichero ; *****THALASSA
if w_device ne '' then begin
  set_plot,w_device
; device,fichero='map.'+w_device,yoffset=(29.-18./prnt)/2.,xsize=18,xoffset=1.5,ys
; Set xoffset/yoffset to default if not specified. (Landscape, Portrait)
  IF NOT KEYWORD_SET(yoffset) THEN yoffset0=[xsz+(29.7-xsz)/2.,(29.-ysz)/2.]$
  ELSE yoffset0= REPLICATE(yoffset, 2)
  IF NOT KEYWORD_SET(xoffset) THEN xoffset0= [(21.-ysz)/2., (21.-xsz)/2.] $
  ELSE xoffset0= REPLICATE(xoffset, 2)

  PRINT, 'Xoff=', xoffset0
  PRINT, 'Yoff=', yoffset0

; Set graphic plot to the specified device.
  IF KEYWORD_SET(landscape) AND landscape EQ 1 THEN BEGIN
    ioff= 0
    DEVICE, filename=out_flg+'.'+w_device, xsize=xsz, ysize=ysz $
      , yoffset=yoffset0(ioff), xoffset= xoffset0(ioff) $
; ; , yoffset= xsz+(29.7-xsz)/2., xoffset= (21.-xsz)/2. $
      , landscape=1
  ENDIF ELSE BEGIN
    ioff= 1
    DEVICE, filename=out_flg+'.'+w_device, ysize=ysz, xsize=xsz $
      , yoffset=yoffset0(ioff), xoffset= xoffset0(ioff) $
; ; , yoffset=(29.-ysz)/2., xoffset=(21.-xsz)/2. $
      , portrait=1
  ENDELSE

  IF w_device EQ 'ps' THEN IF KEYWORD_SET(color) THEN $ ;***COLOR
; Hacer que me ponga en color
;***COLOR
  DEVICE, /color, bits_per_pixel=8
;***COLOR

  call_procedure,temporary,prnt=prnt
```

```
device,/close
```

```
endif  
SET_PLOT, old_pant ; RESET THE OUTPUT DEVICE TO THE WINDOW DEVICE.
```

```
; REMOVE THE TEMPORARY FILE.
```

```
if temporary ne fichero then BEGIN  
    OPENR, untmp, 'xxxxxxxx.pro', /get, /delete  
    FREE_LUN, untmp
```

```
ENDIF
```

```
; Put back the color and background.
```

```
!p.color= 0 & !p.background= !d.n_colors- 1
```

```
END
```

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 1](#) **Up:** [No Title](#) **Previous:** [Contents](#)

List of Figures

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos Utiles](#) **Up:** [No Title](#) **Previous:** [List of Figures](#)

INTRODUCCION A I.D.L 1

(Interactive Data Language)

IDL es un software que se puede utilizar para cálculos simples, programación, representación de gráficos, y procesamiento de imágenes.

-
- [Comandos Utiles:](#)
 - [Datos: tipo y estructura](#)
 - [Cálculos en IDL:](#)
 - [Operaciones simples:](#)
 - [Operaciones mas complejas:](#)
-

1999-03-07

Comandos Utiles:

- **help** nos muestra en pantalla el estado de la sesión de IDL.

ej.: help
nos da todas las informaciones sobre la memoria usada, las subrutinas (procedures) compiladas y las variables en memoria.

help,[1,2]
nos dice que es una expresión formada con numeros enteros y es una matriz con 2 elementos.

- **print** este comando muestra el valor del argumento en pantalla. Se ejecuta con:

PRINT,argumento.

El argumento puede ser un escalar, un vector o una expresión.

ej.: print,2

da

2

1999-03-07

Datos: tipo y estructura

IDL trabaja con 7 tipos de datos de diferentes precisiones, tal y como muestra la tabla posterior:

tipo de datos		bytes		forma		ejemplo
			deci.(*)	hexa.(*)	octa.(*)	
complejo		par de d.p()				complex(n,m)
						complex(n)
doble precision		64	nD			5D
			n.nD			4.5D
			n.nDsx			6.8D-5
real		32	n. .n n.n			5. .6 6.83
			nEsx			5E5
			n.nEsx			54.2E-4
entera	long	32	nL	'n'XL	"nL 'n'OL	dec= 7L
	entera	16	n	'n'X	"n 'n'O	dec= 64
	byte	8	nB	'n'XB	"nB	dec= 34B
caractera alfabética		grupo de 8				'Física'
						"Oceano"
						"it's"

Tabla 1

(*) deci: decimal, hexa: hexadecimal, octa: octal, ()d.p.: doble precision

Los datos se pueden almacenar en una memoria asignándole un nombre (variable). Para hacerlo se utiliza "=". Una variable puede ser de diferente estructura:

- *Escalar* con dimensión = 0.

ej.: x1=6 con dim= 0

x='pal1' con dim= 0

- *Array, Vector, Matriz* con dimensión > 0.

ej.: `y=[5]` con `dim = 1`

`arr=[3,4,5]` con `dim = 1`

`arr2=[[pal1,pal2,pal3],[pal4,pal5,pal6]]` con `dim = 2`
o `dim > 2`.

El nombre de una variable puede contener de 1 a 15 caracteres, con las siguientes restricciones:

(i) El primer carácter tiene que ser una letra del alfabeto.

(ii) No se puede usar los nombres utilizados por IDL, como p.e. nombres de programas, funciones y comandos. Una lista de las palabras de comandos reservados por IDL puede encontrarse en P.3-6U de la guía para IDL.

IDL tiene otro tipo de variables, *variable de sistema*, que son constantes y variables predefinidas. No se puede variar el valor de las constantes asignado inicialmente por IDL, pero sí el de las variables. Estas variables de sistema que se puede variar se usan para controlar gráficas, manejar errores y configurar IDL. Para llamarlas se les antepone "!".

ej.: `!DPI` un constante que da el valor de con doble precisión.

`!PI` el constante con single precisión.

`!p.background` tiene el color del fondo para las gráficas.

`!p.color` tiene el color de las gráficas.

Para ver los valores, ejecute el comando **PRINT** con el argumento que quiere conocer.

NOTA: Por defecto, 255 es el color blanco, y 0 es el color negro.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Cálculos en IDL](#): **Up:** [INTRODUCCION A I.D.L 1](#) **Previous:** [Comandos Utiles:](#)
1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Operaciones simples:](#) **Up:** [INTRODUCCION A I.D.L 1](#) **Previous:** [Datos: tipo y estructura](#)

Cálculos en IDL:

- [Operaciones simples:](#)
 - [Operaciones mas complejas:](#)
-

1999-03-07

Next: [Operaciones mas complejas](#): **Up:** [Cálculos en IDL](#): **Previous:** [Cálculos en IDL](#):

Operaciones simples:

Operadores normales: + - * / #

Operadores relacionales: EQ NE LT GT LE GE

Operadores de Boolean: AND OR NOT XOR.

- Las operaciones se hacen dependiendo del orden de preferencia definido por la tabla 2. Los operadores con la misma jerarquía se realizan de izquierda a derecha.

ej.: `print,5.*2 ^2 da 20` mientras que

`print,(5.*2) ^2 da 100.`

- Para las operaciones con variables de diferente precisión, IDL convierte todos los operandos al tipo de variable que mayor precision haya en la operación.

ej.: `print,5 + 2. da 7.0000`

`print, 8 +2L da a "long" 10`

`print,5/2 + 1. da 3.000` porque primero se hace la división dando un entero = 2, despues al que 1. es un numero real, se convierte a real 2.0000 y se suma con 1.. El resultado real se calcula con

`print,5./2 +1. o`

`print,5/2. + 1 que da 3.5000`

- Las operaciones entre diferentes tipos de "array" se hacen elemento por elemento. Si uno de los "array" tiene menos elementos que los otros, el resultado tendrá lo mismo numero de elementos que el más pequeño.

ej.: `print,[1,2,3]+[-1,-2] da [0,0]`

prioridad	operador
primero	()
segundo	^(exponenciación)
tercero	* (multiplicación)
	# (multiplicación matricial)
	/ (división)
	MOD (módulo)
cuarto	+ (adición)

	- (sustracción)
	< (mínimo)
	> (máximo)
	NOT (negación de boolean)
quinto	EQ (igualdad)
	NE (diferente)
	LE (menor o igual)
	LT (menor)
	GE (más o igual)
	GT (máas)
sexto	AND
	OR XOR

Tabla 2

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Operaciones mas complejas:](#) **Up:** [Cálculos en IDL:](#) **Previous:** [Cálculos en IDL:](#)

1999-03-07

Next: [INTRODUCCION A I.D.L 2](#) **Up:** [Cálculos en IDL](#) **Previous:** [Operaciones simples](#):

Operaciones mas complejas:

Tales como: abs, cos, sin, tan, cosh, sinh, acos, asin, exp, alog, alog10, round, total, etc...Son de tipo función:

$$y=f(x)$$

ej.: print,cos(!PI)

print,exp(23.)

A diferencia de otros programas (fortran, calculadores, etc...) las operaciones en IDL se hacen, tanto si los operandos son escalares como si son matrices o vectores. De esta forma, cualquier operación con un "array" dará como resultado también un "array".

ej.: print,[1,2,3] + 2 da [3,4,5]

x=[2,3]

print,2 ^x da [4,8]

b=3.

print,x*b+1. da [7.0000,10.0000]

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos:](#) **Up:** [No Title](#) **Previous:** [Operaciones mas complejas:](#)

INTRODUCCION A I.D.L 2

- [Comandos:](#)
 - [Array, Vector y Matrices:](#)
 - [Subíndice:](#)
 - [Multiplicación matricial #:](#)
 - [Simple 2-D gráficas:](#)
 - [Grabar sesión en IDL:](#)
 - [Programa simple en IDL:](#)
-

1999-03-07

Next: [Array, Vector y Matrices](#): **Up:** [INTRODUCCION A I.D.L 2](#) **Previous:** [INTRODUCCION A I.D.L 2](#)

Comandos:

- **CD:** para cambiar de directorio de trabajo.

ej.: `cd,'M: }IDL'`

- **PRINTD:** muestra en la pantalla el directorio de trabajo.

ej.: `printd`
da

Current directory: M: }IDL

- **[tipo]INDGEN:** crea un "array " de dimensión definida, donde el valor de cada elemento se corresponde con la posición que dicho elemento ocupa dentro del array. Para ejecutar esta función, haga:

result=[tipo]indgen(D₁[,...,D_n])

[tipo] es para definir el tipo del dato, puede ser:

i : para un array de datos enteros.

b : para un array de byte.

c : para un array de datos complejos.

d : para un array de datos de doble-precisión.

f : para un array de datos reales.

s : para un array de datos formados por caracteres alfabéticos y alfanuméricos.

D_i es el número de elementos de la dimensión que está en la posición *i*.

ej.: `a=indgen(5)`

da un array con los siguientes elementos:

0 1 2 3 4

`b=findgen(4,5)`

da un array de 2 dimensiones con 4 y 5 elementos respectivamente.

- **JOURNAL:** guarda toda la sesión interactiva en un archivo. Para ejecutar este comando, haga:

journal['nombre.pro']

donde *nombre.pro* es el nombre del archivo donde se salva toda la sesión. Cuando quiera terminar de salvar la sesión, ejecute otra vez el comando JOURNAL.

Para cambiar lo que esta gravando, hay que acabar con JOURNAL y utilizar un editor normal.

- **PLOT:** crea una gráfica con 2 dimensiones (x frente a y). Para ejecutarlo, haga:

plot,[X,]Y

donde X es el vector que se representa en el eje de abcisas y Y el vector que se representa en el eje de ordenadas Si X no está presente, la gráfica será el valor de cada Y frente su posición en el vector. La posición empieza con 0.

- **REPLICATE:** crea un array de dimensión definida. Todos los elementos del array tendrán el mismo valor (para cualquier tipo de datos). Para ejecutarlo, haga:

result=replicate(val,D₁[,...,D_n])

donde val es el valor asignado para todo el array. D_i es el número de elemento para cada dimensión i y n es la dimensión del array.

- **SIZE:** esta función da toda la información sobre una variable. El resultado es un vector de números enteros largos con 3 o más elementos.
 - El primer elemento es la dimensión de la variable: 0 para escalar, 1 o más para vector y matriz.
 - El último elemento indica el número de elementos que la variable tiene.
 - El elemento penúltimo indica el tipo de datos que forman la variable (Tabla 3)
 - Dependiendo del valor del primer elemento, habrá 0, 1 o más elementos entre el primero y el penúltimo. Estos elementos dan el número de datos que hay en cada dimensión.

Codigo	tipo de datos
0	indefinido
1	byte
2	entera
3	entera largo (long)
4	real
5	real doble precisión
6	complejo real
7	caracteres alfabéticos y
	alfanuméricos
8	estructura

Tabla.3

ej.: x=5

Comandos:

```
print,size(x)
```

```
da
```

```
0 2 1
```

donde 0 indica escalar, 2 indica datos de tipo entero y 1 indica un solo elemento.

```
y=[1,0]
```

```
print,size(y)
```

```
da
```

```
1 2 2 2
```

```
z=[[0.,1.],[2,3.]]
```

```
print,size(z)
```

```
da
```

```
2 2 2 3 4
```

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Array, Vector y Matrices](#) **Up:** [INTRODUCCION A I.D.L 2](#) **Previous:** [INTRODUCCION A I.D.L 2](#)

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Subíndice:](#) **Up:** [INTRODUCCION A I.D.L 2](#) **Previous:** [Comandos:](#)

Array, Vector y Matrices:

- [Subíndice:](#)
 - [Multiplicación matricial #:](#)
-

1999-03-07

Next: [Multiplicación matricial #](#): **Up:** [Array, Vector y Matrices](#): **Previous:** [Array, Vector y Matrices](#):

Subíndice:

El valor de cada elemento de un array (, vector o matriz) está asignado a un nombre de la variable y a un subíndice. El primer elemento del array tiene el subíndice 0, el subíndice 1 es el elemento que sigue al elemento 0 en la horizontal; y así hasta el final de la primera fila y progresivamente para las siguientes filas.

ej.: El siguiente array

$$\begin{array}{cc} A_{0,0} & A_{1,0} \\ A_{0,1} & A_{1,1} \end{array}$$

se salva como $A_{0,0}, A_{1,0}, A_{0,1}, A_{1,1}$.

IDL permite llamar a uno o más elementos del array utilizando una lista de subíndices. La sintaxis de una referencia de subíndice es:

nombre_de_variable(Lista_de_subíndices)

o

expresión(Lista_de_subíndices)

ej.: array1=[3,2,6,3,8,3,8,4]

print,array1(2)

da

6

array2=[[1,2],[-1,5]]

print,array2(0,1)

da

-1

- Para llamar a más de un elemento de un array, hay que utilizar una lista de números enteros que se pueden guardar en otro array.

ej.: array3=findgen(21)+1

Subíndice:

```
sub=[0,2,4,6]
```

```
print,array3(sub)
```

```
da
```

```
1 3 5 7
```

- Se puede llamar también a un rango de subíndices; la secuencia de llamada es:

```
Array(sb1:sb2)
```

sb1 es el principio del subíndice y sb2 el último. Se usa "*" para llamar todos los elementos o los que son después sb1.

```
ej.: print,array3(14:16)
```

```
da
```

```
14 15 16
```

```
print,array3(19:*)
```

```
da
```

```
19 20 21
```

```
array4=findgen(5,6)
```

```
print,array4(1:2,3:5)
```

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Multiplicación matricial #:](#) **Up:** [Array, Vector y Matrices:](#) **Previous:** [Array, Vector y Matrices:](#)
1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Simple 2-D gráficas:](#) **Up:** [Array, Vector y Matrices:](#) **Previous:** [Subíndice:](#)

Multiplicación matricial #:

Las características y restricciones para este comando son:

- La segunda dimensión del primer operando debe ser lo mismo que la primera dimensión del segundo operando. $A(n,m) \# B(m,p)$
- El resultado tendrá su primera dimensión igual a la primera dimensión del primer operando y su segunda dimensión igual a la segunda dimensión del segundo operando. $C(n,p)$

ej.: `array1=indgen(2,5)`

`ar=array3#array4`

esta multiplicación da un array 2 x 6. Para verificarlo, ejecute:

```
print,ar
```

```
o
```

```
print,size(ar)
```

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Grabar sesión en IDL](#): **Up:** [INTRODUCCION A I.D.L 2](#) **Previous:** [Multiplicación matricial #:](#)

Simple 2-D gráficas:

ej.: `plot,array3`

hace una gráfica de `array3` frente la posición de cada elemento del array.

```
array1=sin(2.*!pi*findgen(41)/40.)
```

`plot,array1`

hace una gráfica de seno frente 0-40

```
x=findgen(2,21)
```

```
plot,x(0,*),x(1,*)
```

crea una gráfica de los elementos de la primera columna frente los de la segunda columna.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Programa simple en IDL:](#) **Up:** [INTRODUCCION A I.D.L 2](#) **Previous:** [Simple 2-D gráficas:](#)

Grabar sesión en IDL:

Ahora, asegúrese que está en el directorio M:} y repita el mismo ejemplo, pero primero ejecute el comando:

```
journal,'prueba.pro'
```

Al final ejecute otra vez **JOURNAL** sin poner ninguno argumento.

Puede ver toda la sesión si edite el archivo *prueba.pro*. (utilize un editor normal de DOS, de Window o de IDL)

-
- [Programa simple en IDL:](#)
-

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 3](#) **Up:** [Grabar sesión en IDL:](#) **Previous:** [Grabar sesión en IDL:](#)

Programa simple en IDL:

Si no hay error en la ejecución de cada comando de *prueba.pro*, ya tenemos un simple programa en IDL. Falta solo introducir el comando **END** al final de todo (este comando es necesario para acabar la ejecución del programa).

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos:](#) **Up:** [No Title](#) **Previous:** [Programa simple en IDL:](#)

INTRODUCCION A I.D.L 3

- [Comandos:](#)
 - [Opciones para comandos de gráficas:](#)
 - [Programación:](#)
 - [Inicialización:](#)
 - [Procesamiento:](#)
 - [Gráficos:](#)
 - [Fin de programa:](#)
 - [Ejecución de un programa en IDL:](#)
-

1999-03-07

Comandos:

- **[tipo]ARR:** crea un vector o matriz de dimensión definida. A diferencia del comando *[tipo]indgen*, este comando define la matriz sin asignar valor a los elementos. Para ejecutarla, haga:

result=[tipo]arr(D₁[,...,D_n])

[tipo] es para definir el tipo de dato, puede ser:

- byt** para un array de datos de tipo byte
- complex** para un array de datos de tipo complejo
- dbl** para un array de datos de tipo doble-precisión
- flt** para un array de datos de tipo real
- int** para un array de datos de tipo entero
- lon** para un array de datos de tipo *long* (largo)
- str** para un array de datos de tipo

D_i es el número de elementos de la dimensión que está en la posición *i*

ej.: **a=fltarr(6,20)**

da un matriz de 6x20.

- **CLOSE:** cierra un archivo. Siempre hay que cerrar un archivo antes de abrirlo de nuevo. Se ejecuta con:

close, unidad

Si quiere cerrar todos los archivos, utilice:

close,/all

- **MIN, MAX:** son las funciones que dan el mínimo y el máximo respectivamente de un vector o matriz. Las secuencias de llamada son:

m=min(X[,max=mx])

m=max(X[,min=mn])

Donde *X* es un vector o matriz, *mx* (*mn*) es la variable donde se va a guardar el máximo (o mínimo).

Comandos:

ej.: print, min(a, max=mx), mx

- **OPEN[R,W,U]** abre un archivo de datos. Las opciones son:

R para solamente leer datos.

W para solamente escribir datos

U para escribir y leer datos.

Se ejecuta con:

openr,unidad,' nombre_del_archivo'

openw,unidad,' nombre_del_archivo'

openu,unidad,' nombre_del_archivo'

unidad es el número entero que se asigna al archivo.

ej.: openr,1,'datos1.dat'

- **PRINTF:** hace lo mismo que PRINT, pero en vez de escribir el resultado en la pantalla, lo escribe en un archivo. La secuencia de llamada es:

PRINTF,unidad,datos1[,daots2,...]

Donde *unidad* especifica el numero asignado al archivo. No se olvide de abrir el archivo antes de escribir nada.

- **RANDOMU:** es una función que genera un vector o matriz de números aleatorios reales entre 0 y 1. La secuencia de llamada es:

a=randomu(m[,D_i,...,D_n])

Donde *m* es una variable que se genera automáticamente, y *D_i* las dimensiones. Si no se especifica ninguna dimensión, el resultado será un escalar.

ej.: a=randomu(s,5)

b=randomu(s,11,16)

- **READ:** lee los valores escritos en la pantalla (introducidos por medio del teclado). Para ejecutarlo, haga:

read,a[,b,...]

a[,b,...] son los nombres de las variables donde se van a almacenar los valores. Por defecto, el comando lee un número entero. Este comando tiene varias opciones tales como **format**, y **prompt**.

- **READF:** lee los valores contenidos en un archivo. Para ejecutarlo:

readf,unidad,a1[,a2,...]

unidad es el número asignado para el archivo.

Comandos:

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Opciones para comandos de I.D.L 3](#) **Up:** [INTRODUCCION A I.D.L 3](#) **Previous:** [INTRODUCCION A I.D.L 3](#)

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Programación](#): **Up:** [INTRODUCCION A I.D.L 3](#) **Previous:** [Comandos](#):

Opciones para comandos de gráficas:

En general, los comandos de gráficas hacen gráficas simples. Para cambiar la salida de las gráficas, hay que introducir opciones tal como título, tamaño de línea, etc...

Para escribir las opciones:

Para los comandos, las opciones se añaden al final de la sentencia separándolas con una coma (,). Para las funciones, las opciones se introducen dentro de los mismos paréntesis que los argumentos separándolas con coma.

ej.: `plot,x,title='numero aleatorio'` (sin acento para el IDL)

`a=min(x,max=m)`

Algunas de las opciones más utilizadas en las gráficas son:

- **[xyz]title:** para especificar el título de la gráfica, el eje X, el eje Y y el eje Z, respectivamente.
- **linestyle:** para especificar el estilo de la línea.
- **thick:** para especificar el tamaño de la línea.
- **psym:** para especificar si quiere utilizar un signo en cada punto de los datos. Esta opción no se puede combinar con *linestyle*.
- **[xyz]charsize:** para especificar el tamaño de los caracteres escritos en la gráfica para la gráfica, el eje X, el eje Y y el eje Z, respectivamente.
- **color:** para especificar el color de cada línea. Por defecto, los colores son blanco-gris-negro.

Para ver los efectos de estas opciones, mire en la tabla 4.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Programación](#): **Up:** [INTRODUCCION A I.D.L 3](#) **Previous:** [Comandos](#):

1999-03-07

Programación:

Inicialización

Procesamiento => Resultado

Gráficos

fin de programa

-
- [Inicialización:](#)
 - [Procesamiento:](#)
 - [Gráficos:](#)
 - [Fin de programa:](#)
 - [Ejecución de un programa en IDL:](#)
-

1999-03-07

Next: [Procesamiento](#): **Up:** [Programación](#): **Previous:** [Programación](#):

Inicialización:

Hay 2 métodos de inicializar las variables:

1. Asignar valores con comandos

Simple asignación

ej.: a=[1,2,3,4]

Introducir datos desde el teclado

ej.: read,a1,a2,b1,b2

2. Almacenar datos en un archivo

Definir una matriz

Abrir el archivo

Leer y asignar los datos en la matriz

Cerrar el archivo

1999-03-07

Procesamiento:

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Gráficos:](#) **Up:** [Programación:](#) **Previous:** [Inicialización:](#)

Procesamiento:

[Hacer los cálculos necesarios para obtener el resultado.]

1999-03-07

Next: [Fin de programa:](#) **Up:** [Programación:](#) **Previous:** [Procesamiento:](#)

Gráficos:

Se puede hacer gráficas con 2 dimensiones (p.e. plot, contour) o con 3 dimensiones (p.e. surface, shade_surf).

Como ejemplo, vamos a escribir un programa que lea un archivo de datos:

1. Cree un archivo de datos que tenga 4 columnas y 8 filas.
2. Cree un programa que lea los datos dentro del archivo.
3. Haga la gráfica de la columna 0 frente a la columna 3.

En la realidad, tenemos varios archivos de datos y muchas veces queremos elegir el archivo a leer. Para ello, escribimos el nombre del archivo en pantalla.

1. Ahora, cree otro archivo de datos de 4 columnas y 8 filas.
2. Modifique el programa para que puede leer el nombre del archivo desde el teclado (pantalla).
3. Haga la misma gráfica pero con el nuevo dato.
4. Modifique el programa, introduciendo las opciones de título (para la gráfica y para los ejes).
5. Puede elegir también el formato de línea o su tamaño.

1999-03-07

Fin de programa:

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Ejecución de un programa](#) **Up:** [Programación](#) **Previous:** [Gráficos](#):

Fin de programa:

Al final del programa, hay que poner un **END**, que especifica el fin de la ejecución.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 4](#) **Up:** [Programación](#): **Previous:** [Fin de programa](#):

Ejecución de un programa en IDL:

Para ejecutar un programa en IDL, hay que compilarlo y luego ejecutarlo. En el caso más simple, estas 2 acciones se hacen a la vez utilizando:

.RUN (.R)

.RNEW (.RN)

nombre_del_programa (sin la extensión)

A diferencia del primero comando, el segundo comando (.rn) borra primero todas las variables grabadas en la memoria de IDL antes de compilar y ejecutar el programa *nombre_del_programa*.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos](#) **Up:** [No Title](#) **Previous:** [Ejecución de un programa](#)

INTRODUCCION A I.D.L 4

- [Comandos](#):
 - [Bloque grupo de comandos](#):
-

1999-03-07

Next: [Bloque grupo de comandos:](#) **Up:** [INTRODUCCION A I.D.L 4](#) **Previous:** [INTRODUCCION A I.D.L 4](#)

Comandos:

- Para convertir un data de un tipo a otro, utilice los comandos del tabla5:

Función	Descripción
STRING	converte en tipo caractere
BYTE	converte en tipo byte
FIX	converte en tipo (16-bit) entero
LONG	converte en tipo (32-bit) entero
FLOAT	converte en tipo real
DOUBLE	converte en tipo doble-precision
COMPLEX	converte en tipo complejo

Tabla 5.

- **CONTOUR:** hace una gráfica de isolíneas a partir de una matriz de 2 dimensiones. Los datos son localizados en una malla regular. Se ejecuta con:

contour,Z[,X,Y]

Donde Z es una matriz con 2 dimensiones. X e Y son vectores (1 dim.) o matrices (2 dim.) que contienen respectivamente la posición en X e Y de los datos.

Si X e Y no existen, la gráfica se hace con Z frente a la posición de cada elemento dentro de la matriz.

ej.: z=randomu(21,26)

contour,z

x=findgen(21)*1.5

y=findgen(26)*2.

contour,z,x,y

- **SURFACE:** hace una gráfica de 3 dimensiones (Z frentes a X e Y) a partir de datos localizados dentro de una malla regular. La secuencia de llamada es:

surface,Z[,X,Y]

Donde Z es una matriz con 2 dimensiones. X e Y son vectores (1 dim.) o matrices (2 dim.) que contienen

Comandos:

respectivamente la posición en X e Y de los datos.

Si X e Y no existen, la gráfica se hace con Z frente a la posición de cada elemento dentro de la matriz.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Bloque grupo de comandos](#): **Up:** [INTRODUCCION A I.D.L 4](#) **Previous:** [INTRODUCCION A I.D.L 4](#)

1999-03-07

Bloque grupo de comandos:

El block es un grupo de comandos que se considera como un solo comando. Se utiliza cuando se ejecuta este grupo dependiendo de una condición o repitiéndose unas veces.

Para definirlo, el grupo de comandos debe estar dentro de los comandos: BEGIN y END:

BEGIN

comando 1

comando 2

....

comando n

END

Los comandos para verificar una condición son los siguientes:

CASE ENDCASE

IF ENDIF

Los comandos para repetir un grupo de comandos son:

FOR ENDFOR

REPEAT ENDREP

WHILE ENDWHILE

Existen otros comandos de programación que IDL utiliza tales como:

COMMON para definir un grupo de parámetros asignados a unas variables que mas de un programa utilizaran

GOTO salta a la línea que tiene la etiqueta especificada.

Para comprender la utilización de estos comandos, véase los ejemplos.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos](#) **Up:** [No Title](#) **Previous:** [Bloque grupo de comandos:](#)

INTRODUCCION A I.D.L 5

- [Comandos](#)
 - [Caracteres usados para los programas:](#)
 - [Subrutina y Función:](#)
-

1999-03-07

Next: [Caracteres usados para los](#) **Up:** [INTRODUCCION A I.D.L 5](#) **Previous:** [INTRODUCCION A I.D.L 5](#)

Comandos

- **DEVICE:** es el comando para definir las propiedades del dispositivo de gráfica. Para ver las opciones de DEVICE, véanse las ayudas o el manual.
- **FINDFILE:** es una función para ver la lista de los archivos en un directorio. Se ejecuta con:

resultado=findfile(filtro)

donde, filtro es la propiedad de los archivos a listar.

ej.: print,findfile('*pro')

- **HELP,/DEVICE** muestra toda la información sobre el dispositivo de gráfica.
- **OPLOT:** (Over PLOT) lo mismo que PLOT, pero no genera ejes nuevos, este comando muestra la gráfica por encima de una gráfica ya hecha. La secuencia de llamada es:

oplot[,X],Y

donde X es el vector que se representa en el eje de abcisas e Y el vector que se representa en el eje de ordenadas.

- **SET_PLOT:** define el dispositivo (pantalla, fichero, ordenador) para enseñar las gráficas. IDL puede generar sus gráficas en diferentes tipos de dispositivos. Los dispositivos que utilizamos frecuentemente son:

HP Hewlett-Packard Graphics Language (HP-GL) para *plotter*.

NULL sin salida.

PCL Hewlett-Packard Printer Control Language utilizado por las impresoras HP.

PS Postscript.

WIN Pantalla de Window (P.C).

X Pantalla de SUN (X-Windows).

- **WINDOW:** para abrir una pantalla de gráfica. Este comando tiene varias opciones tales como: el numero de la ventana, su tamaño, su posición en la pantalla, etc...

Su tamaño se especifica con las opciones: xsize, ysize. Se ejecuta con:

window[,numero[, xsize=tam_x,ysize=tam_y],...]

Next: [Caracteres usados para los](#) **Up:** [INTRODUCCION A I.D.L 5](#) **Previous:** [INTRODUCCION A I.D.L 5](#)

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Subrutina y Función:](#) **Up:** [INTRODUCCION A I.D.L 5](#) **Previous:** [Comandos](#)

Caracteres usados para los programas:

- Cuando una línea de comando es muy larga, se puede escribir en más de una línea, para eso, se usa la carácter "\$" al final de la línea. Este carácter especifica que la línea siguiente es parte del comando donde se pone el \$.

ej.: `plot,x,y,title='TITULO 1',xrange=[0,14],$`

`yrange=[4,54],xstyle=1,ystyle=1,$`

`xtitle='Eje X',ytitle='Eje Y'`

- Se pueden poner 2 comandos en una línea separándolos con "&".

ej.: `x=[1,2,3] & y=[4,3,2]`

1999-03-07

Next: [INTRODUCCION A I.D.L 6](#) **Up:** [INTRODUCCION A I.D.L 5](#) **Previous:** [Caracteres usados para los](#)

Subrutina y Función:

Los programas se pueden llamar como subrutina o función. En este caso, estas subrutinas y funciones se ejecutan de la misma manera que los comandos y las funciones predefinidos por IDL.

El usuario puede escribir su propia subrutina o función. A diferencia de un programa normal, este tipo de programa debe estar dentro de:

PRO nombre_de_la_subrutina[,param1_de_entrada,...,param1_de_salida,...]

comando 1.

.....

RETURN

END

para la subrutina y dentro de

FUNCTION nombre_de_la_función[,parámetro_1_de_entrada,...]

comando 1

.....

RETURN, param 1_de_salida[, param 2_de_salida,]

END

para las funciones.

Debería llamarlas de la misma manera que el nombre del archivo donde se guarda el programa. Hay que compilarla antes de ejecutarla.

NOTA: **.RUN (.R)** o **.RNEW (.RN)** compila sólo las subrutinas o las funciones sin ejecutarlas.

Para comprender la utilización de las subrutinas y funciones, véase los ejemplos.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Comandos](#) **Up:** [No Title](#) **Previous:** [Subrutina y Función:](#)

INTRODUCCION A I.D.L 6

- [Comandos](#)
 - [Ejercicios para programación en IDL](#)
 - [Objetivo 1: *Hacer una gráfica donde se superponen diferentes líneas con colores diferentes al del borde de la gráfica.*](#)
 - [Objetivo 2: *Comprender la idea de subrutina/función con argumentos y opciones.*](#)
-

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Ejercicios para programación en](#) **Up:** [INTRODUCCION A I.D.L 6](#) **Previous:** [INTRODUCCION A I.D.L 6](#)

Comandos

- **CURSOR:** Se usa para ver la posición del cursor en una ventana de gráfica. Se ejecuta con el comando:
 . **cursor,x,y,data=1**
 , y luego hay que pulsar el botón de la izquierda del ratón. El cursor debe estar en una ventana de gráfica cuando pulsa el botón del ratón.
- **LOADCT:** Este comando da las opciones de colores que IDL tiene predefinidas. Se ejecuta solo, sin argumentos posteriores, y salen por pantalla las opciones que podemos elegir.
- **STR.....:** Los comandos que empiezan con STR..son funciones para caracteres, p.e. STRPOS es una función útil para buscar una palabra dentro de un grupo de caracteres.
- **XYOUTS:** es el comando para escribir caracteres en la pantalla de gráfica. La secuencia de llamada es:
 . **xyouts,x_pos,y_pos,'palabra'[,opciones,....]**

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo 1:Hacer una gráfica](#) **Up:** [INTRODUCCION A I.D.L 6](#) **Previous:** [Comandos](#)

Ejercicios para programación en IDL

- [Objetivo 1:Hacer una gráfica donde se superponen diferentes líneas con colores diferentes al del borde de la gráfica.](#)
 - [Objetivo 2:Comprender la idea de subrutina/función con argumentos y opciones.](#)
-

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo 2:Comprender la idea](#) **Up:** [Ejercicios para programación en](#) **Previous:** [Ejercicios para programación en](#)

Objetivo 1:Hacer una gráfica donde se superponen diferentes líneas con colores diferentes al del borde de la gráfica.

Para esto, el algoritmo a escribir es un programa que:

1. Lea un archivo de datos. p.e. 5x51. (x,y)
2. Haga una gráfica sin datos, solo el borde, utilizando la opción: **NODATA=1** y especificando el color asignado al borde.
3. Represente la primera curva con otro color utilizando **OPlot**.
4. Pueda también superponer más curvas en la misma gráfica.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 7](#) **Up:** [Ejercicios para programación en](#) **Previous:** [Objetivo 1:Hacer una gráfica](#)

Objetivo 2:Comprender la idea de subrutina/función con argumentos y opciones.

1.

Para esto, escribiremos un primer programa cuyos pasos son:

- (a) Definir las variables. p.e. 5x51
- (b) Preguntar el nombre del archivo de datos.
- (c) leer los datos X.
- (d) Preguntar el número de la columna que se quiere representar.
- (e) Calcular la función $\text{COS} + 1$. de la columna especificada, y grabar el resultado en otra variable Y.
- (f) Hacer la gráfica de la columna 0 de X frente a Y.
Grabe este programa en un archivo y ejecútelo. No se olvide que .RUN (.RNEW) compilan y ejecutan un programa principal. Entonces, sería:

.RNEW nombre_del_programa

2.

Cambie este programa en una subrutina. Para ello, ponga

. **PRO nb_del_prog** al principio y
. **RETURN** antes del **END** del programa.

Guarde este nuevo programa en otro archivo que tenga el mismo nombre que el programa. En este caso, **.RNEW (.RUN)** solo compila sin ejecutar el programa. Para ejecutarlo, luego debe escribir el nombre del programa y pulsar la tecla *INTRO* (sin **.RUN**).

. **.RNEW nb_de_la_sub** para compilar y
. **nb_de_la_sub** para ejecutarla.

3.

Nuevo cambio: borre la parte que pide el número de columna y ponga el nombre de la variable detrás del nombre del programa separándolos con "," (coma).

Grabe este nuevo programa en otro archivo y compílelo.

Para ejecutar este programa, recordemos que, ahora, número de columna es un argumento de la subrutina, esto significa que tenemos que especificarlo cuando ejecutamos el programa (subrutina). Se ejecuta con:

nb_de_la_sub,3

4.

Otra alternativa es utilizar la columna número 2 por defecto. Para eso, cambiamos la primera línea en:

. PRO nb_del_prog,col=col

e introducimos el comando:

. IF n_elements(col) eq 0 then col=2

Grabe este programa y compílelo. Para ejecutarlo, puede especificar la columna con:

. nb_del_prog,colo=3

o dejar IDL utilizar el defecto.

5.

Por último, escriba un programa con la función que calcula la COS +1. de un vector o matriz. Los pasos son:

(a)

Ponga

. FUNCTION nb_de_la_función,x

al principio del programa. X es el argumento.

(b)

Ponga

. RETURN,cos(x)

antes de **END**.

Grábelo en un archivo que tenga el mismo nombre que la función, pero con extensión .PRO. Compile esta función. Para ejecutarla, puede llamarla desde la subrutina

. Y=nb_de_la_función(X(i,*))

en lugar de **Y=cos(X(i,*))**.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 7](#) **Up:** [Ejercicios para programación en](#) **Previous:** [Objetivo 1:Hacer una gráfica](#)

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo n1: Gráfica de](#) **Up:** [No Title](#) **Previous:** [Objetivo 2:Comprender la idea](#)

INTRODUCCION A I.D.L 7

- [Objetivo n1: Gráfica de isolínea con datos repartidos en una malla irregular:](#)
 - [Objetivo n2: Creación de nuevo eje:](#)
 - [Objetivo n3: Rellenar isolíneas con color o banda:](#)
 - [Objetivo n4: Utilización del comando CURSOR:](#)
-

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo n2: Creación de](#) Up: [INTRODUCCION A I.D.L 7](#) Previous: [INTRODUCCION A I.D.L 7](#)

Objetivo n1: Gráfica de isolínea con datos repartidos en una malla irregular:

1.

Ejecute el programa: **IRR_GRID.PRO**

2.

El programa pregunte si quiere seguir o acabar, conteste con " a" para acabar la ejecución. Hay 2 ventanas de gráfica, n 0 y n 1. Deberíamos ver la ventana n1, si no es el caso, ejecute el comando:

wshow,1

3.

Escribe en la pantalla los valores de cada punto con su posición. Para eso, ejecute el comando:
for i=0,24 do print,i,x(i),y(i),z(i)

4.

Ahora, mire la posición del *cursor* utilizando el programa: **CURS_POS.PRO**

5.

Intente ver en que posición corresponde al punto elegido, y compare el valor de la gráfica con los datos.

6.

Estudio del programa:

(a)

La copia del programa tiene unos comentarios que nos ayudaran a comprender cada paso.

(b)

Además, hay espacio para poner más comentario si se quiere.

(c)

Mire el tipo de variables utilizadas. Para eso, ejecute el comando:

help,x,y,z,xx,yy,zz

(d)

Se ve que *zz* es una matriz con 2 dimensiones. Esto es necesario porque los comandos *contour*, *surface*, *shade_surf* trabajan con matrices de 2 dimensiones.

(e)

En el programa, se especifica que los 11 niveles de isolíneas son repartidos entre:

Objetivo n1: Gráfica de isolínea con datos repartidos en una malla irregular:

0 y $\max(z)$ para la gráfica en la ventana n0

$\min(z)$ y $\max(z)$ para la 2da gráfica.

La diferencia viene de los comandos

zz(where(zz eq 0))=1000

.....

zmx=max(z,min=zmn)

contour, zz,...,max_value=zmx+0.1

Esto, porque IDL asigna un valor 0 para los puntos donde no hay dato. Para asegurarse del valor de punto donde no hay datos, puede ejecutar el programa

IRR_SURF.PRO

(f)

También, observe el cambio de los valores de los ejes. En la segunda gráfica, los ejes son definidos ejecutando las líneas de comando:

xx=findgen(...

yy=findgen(...

que crean 2 vectores con lo mismo número de elemento que las dimensiones de la matriz zz.

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo n2: Creación de](#) **Up:** [INTRODUCCION A I.D.L 7](#) **Previous:** [INTRODUCCION A I.D.L](#)

[7](#)

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo n3: Rellenar isolíneas](#) **Up:** [INTRODUCCION A I.D.L 7](#) **Previous:** [Objetivo n1: Gráfica de](#)

Objetivo n2: Creación de nuevo eje:

1.

Ahora, ejecute otra vez el programa

IRR_GRID.PRO

y acaba contestando con "a" a la pregunta. Ejecute el comando

axis,0.5,0.5,xaxis=1,xrange=[0,1]*10

Este comando genera un eje X. Para ver como se usa este comando, mire en la ayuda de IDL.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [Objetivo n4: Utilización del](#) **Up:** [INTRODUCCION A I.D.L 7](#) **Previous:** [Objetivo n2: Creación de](#)

Objetivo n3: Rellenar isolíneas con color o banda:

1.

Ejecutando el primer programa y contestando con "s", puede ver como se rellena las isolíneas con color o con banda.

NOTA: No se puede rellena las isolíneas que no se cierran.

2.

El programa: **FILL_CNT.PRO** muestra otro ejemplo de este tipo.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [INTRODUCCION A I.D.L 8](#) **Up:** [INTRODUCCION A I.D.L 7](#) **Previous:** [Objetivo n3: Rellenar isolíneas](#)

Objetivo n4: Utilización del comando CURSOR:

Este comando se utiliza para saber la posición del cursor en la ventana de gráfica. Para verlo,

1.

Crea una gráfica con
plot,[2,4,6,2,7,9,2,5,2]

2.

ejecute el comando
CURS_POS.PRO
Este programa muestra la posición del cursor en la ventana.

3.

Borra la gráfica utilizando el comando
erase

4.

Ejecute el programa
DRAW_GRF.PRO

5.

Puede empezar a dibujar, en cada punto debe pulsar en el botón de la izquierda, para acabar el dibujo, pulsa el botón de la derecha.

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Next: [About this document ...](#) **Up:** [No Title](#) **Previous:** [Objetivo n4: Utilización del](#)

INTRODUCCION A I.D.L 8

1999-03-07

[Next](#) [Up](#) [Previous](#) [Contents](#)

Up: [No Title](#) Previous: [INTRODUCCION A I.D.L 8](#)

About this document ...

This document was generated using the [LaTeX2HTML](#) translator Version 98.1p1 release (March 2nd, 1998)

Copyright © 1993, 1994, 1995, 1996, 1997, [Nikos Drakos](#), Computer Based Learning Unit, University of Leeds.

The command line arguments were:

latex2html class_sp.tex.

The translation was initiated by [Andry](#) on 1999-03-07

1999-03-07