

GDL – GNU Data Language

a free/libre/open-source implementation of IDL/PV-WAVE*

developed by Marc Schellens and The GDL team
documentation maintained by Sylwester Arabas and Alain Coulais

January 3, 2012

* IDL (Interactive Data Language) and PV-WAVE (Precision Visuals - Workstation Analysis and Visualization Environment) are (were) registered trademarks of EXELIS VIS (ITT VIS; Research Systems, Inc.) and Rogue Wave Software (Visual Numerics; Precision Visuals), respectively

Contents

About GDL	7	Chapter 3. Interpreter commands and built-in debugging facilities	17	Misc	24
License	7	Chapter 4. Maths	18	Chapter 7. Interaction with host OS	25
Credits	7	Basic Scalar, vector and array operations	18	Executing external commands (via shell or not)	25
Providing feedback	7	Basic and special function library	18	Filesystem operations	25
Organization of this document	8	Linear algebra	19	Network operations	25
		Statistics	19	Command-line options and environmental variables	25
		Interpolation	19	Chapter 8. Manipulating strings	26
		Polynomials	19	Chapter 9. Representing date & time	27
		Geometric calculations	19	Chapter 10. Image processing	28
		Bitwise operations	19	Chapter 11. Parallel processing	29
		Function fitting	19	Built-in features (OpenMP)	29
		Fourier analysis	20	Semaphores and shared memory (library routines)	29
		Multidimensional root-finding	20	ImageMagick's features	29
		Random numbers	20	MPI and GDL	29
		Ordinary differential equations	20	Chapter 12. GUI programming (widgets)	30
		Wavelet analysis	20	Chapter 13. Dynamic loading	31
		Mathematical and physical constants	20	Chapter 14. The Python bridge	32
Chapter 1. Obtaining, installing, and invoking GDL	11	Chapter 5. Input/output, supported data formats	21	calling Python code from GDL	32
Requirements and supported environments	11	Basics – accessing files and io streams	21	calling GDL code from Python	32
Availability of pre-compiled packages	11	ASCII	21	Chapter 15. Alphabetical list of library routines	33
Compiling GDL from source	11	CSV	21	ABS() function	33
Installation layout	11	Binary data (raw access)	21	ACOS() function	33
Command-line options	11	FITS	21	ALOG() function	33
Influential environmental variables	11	netCDF	21	ALOG10() function	33
Chapter 2. Language reference	12	HDF4	21	APPLEMAN procedure	33
Syntax basics	12	HDF5	21	ARG_PRESENT() function	34
Datatypes	12	raster images (TIFF, PNG, JPEG, ...)	22	ARRAY_EQUAL() function	34
Operators	12	DICOM	22	ARRAY_INDICES() function	34
Flow control structures	12	GRIB	22	ASIN() function	34
Variable scoping rules	14	IDL save files	22	ASSOC() function	34
Functions and procedures	14	Chapter 6. Plotting and mapping	23	ATAN() function	34
Argument passing	14	2D plots	23	AXIS procedure	35
Arrays	15	3D plots	23	BESEL() function	35
Structures	15	Plotting raster data	23	BESELJ() function	35
System variables (global)	15	Managing multiple windows	23	BESELK() function	35
Heap variables (pointers)	15	Map projections	23	BESELY() function	35
The HELP procedure	15	Output terminals	23	BETA() function	35
Object-oriented programming	15	Working with colours	23	BILINEAR() function	35
Handling Overflows, Floating Point Special Values	15	Fonts, symbols and text formatting	24		
Error handling	15				
Compile options	15				

BINDGEN() function	35	DETERM() function	41	FLUSH procedure	46
BROYDEN() function	35	DEVICE procedure	41	FREE_LUN procedure	46
BYTARR() function	35	DIALOG_MESSAGE() function	42	FSTAT() function	46
BYTE() function	35	DIALOG_PICKFILE() function	42	GAMMA() function	46
BYTEORDER procedure	36	DINDGEN() function	42	GAUSSINT() function	46
BYTSCL() function	36	DIST() function	42	GAUSS_CVF() function	46
CALDAT procedure	36	DOUBLE() function	42	GAUSS_PDF() function	46
CALENDAR procedure	36	EOF() function	42	GDL_ERFINV() function	47
CALL_EXTERNAL() function	36	ERASE procedure	42	GETENV() function	47
CALL_FUNCTION() function	38	ERF() function	42	GET_DRIVE_LIST() function	47
CALL_METHOD procedure	38	ERFC() function	43	GET_KBRD() function	47
CALL_METHOD() function	38	ERRORF() function	43	GET_LOGIN_INFO() function	47
CALL_PROCEDURE procedure	38	ESCAPE_SPECIAL_CHAR() function	43	GET_LUN procedure	47
CATCH procedure	38	EXECUTE() function	43	GET_SCREEN_SIZE() function	47
CD procedure	38	EXIT procedure	43	GRIBAPI_CLONE() function	47
CDF_EPOCH procedure	38	EXP() function	43	GRIBAPI_CLOSE_FILE procedure	47
CEIL() function	39	EXPAND_PATH() function	43	GRIBAPI_COUNT_IN_FILE() function	47
CHECK_MATH() function	39	EXPINT() function	43	GRIBAPI_GET procedure	47
CINDGEN() function	39	FACTORIAL() function	43	GRIBAPI_GET_DATA procedure	47
CLOSE procedure	39	FFT() function	44	GRIBAPI_GET_SIZE() function	48
COMMAND_LINE_ARGS() function	39	FILEPATH() function	44	GRIBAPI_NEW_FROM_FILE() function	48
COMPLEX() function	39	FILE_BASENAME() function	44	GRIBAPI_OPEN_FILE() function	48
COMPLEXARR() function	39	FILE_COPY procedure	44	GRIBAPI_RELEASE procedure	48
CONGRID() function	39	FILE_DELETE procedure	44	GSL_EXP() function	48
CONJ() function	39	FILE_DIRNAME() function	45	H5A_CLOSE procedure	48
CONTOUR procedure	39	FILE_EXPAND_PATH() function	45	H5A_GET_NAME() function	48
CONVERT_COORD() function	39	FILE_INFO() function	45	H5A_GET_NUM_ATTRS() function	48
CONVOL() function	40	FILE_LINES() function	45	H5A_GET_SPACE() function	48
CORRELATE() function	40	FILE_MKDIR procedure	45	H5A_GET_TYPE() function	48
COS() function	40	FILE_SAME() function	45	H5A_OPEN_IDX() function	48
COSH() function	40	FILE_SEARCH() function	45	H5A_OPEN_NAME() function	48
CPU procedure	41	FILE_TEST() function	45	H5A_READ() function	48
CREATE_STRUCT() function	41	FILE_WHICH() function	45	H5D_CLOSE procedure	48
CROSSP() function	41	FINDEX() function	45	H5D_GET_SPACE() function	49
CURSOR procedure	41	FINDFILE() function	45	H5D_GET_TYPE() function	49
DBLARR() function	41	FINDGEN() function	46	H5D_OPEN() function	49
DCINDGEN() function	41	FINITE() function	46	H5D_READ() function	49
DCOMPLEX() function	41	FIX() function	46	H5F_CLOSE procedure	49
DCOMPLEXARR() function	41	FLOAT() function	46	H5F_IS_HDF5() function	49
DEFSYSV procedure	41	FLOOR() function	46	H5F_OPEN() function	49
DERIV() function	41	FLTARR() function	46	H5G_CLOSE procedure	49

H5G_OPEN() function	49	IGAMMA() function	53	MAGICK_CLOSE procedure	59
H5S_CLOSE procedure	49	IMAGE_STATISTICS procedure	53	MAGICK_COLORMAPSIZE() function	60
H5S_GET_SIMPLE_EXTENT_DIMS() function	49	IMAGINARY() function	53	MAGICK_COLUMNS() function	60
H5T_CLOSE procedure	49	IMSL_BINOMIALCOEF() function	53	MAGICK_CREATE() function	60
H5T_GET_SIZE() function	49	IMSL_CONSTANT() function	54	MAGICK_DISPLAY procedure	60
H5_GET_LIBVERSION() function	50	IMSL_ERF() function	55	MAGICK_EXISTS() function	60
HDF_CLOSE procedure	50	IMSL_ZEROPOLY() function	55	MAGICK_FLIP procedure	60
HDF_OPEN() function	50	IMSL_ZEROSYS() function	55	MAGICK_INDEXEDCOLOR() function	60
HDF_SD_ADDDATA procedure	50	INDGEN() function	56	MAGICK_INTERLACE procedure	60
HDF_SD_ATTRFIND() function	50	INTARR() function	56	MAGICK_MAGICK() function	60
HDF_SD_ATTRINFO procedure	50	INTERPOL() function	56	MAGICK_MATTE procedure	60
HDF_SD_CREATE() function	50	INTERPOLATE() function	56	MAGICK_OPEN() function	60
HDF_SD_DIMGET procedure	50	INVERT() function	56	MAGICK_PING() function	60
HDF_SD_DIMGETID() function	50	ISHFT() function	56	MAGICK_QUALITY procedure	60
HDF_SD_END procedure	50	JOURNAL procedure	56	MAGICK_QUANTIZE procedure	61
HDF_SD_ENDACCESS procedure	50	KEYWORD_SET() function	56	MAGICK_READ() function	61
HDF_SD_FILEINFO procedure	50	KURTOSIS() function	56	MAGICK_READCOLORMAPRGB procedure	61
HDF_SD_GETDATA procedure	51	L64INDGEN() function	56	MAGICK_READINDEXES() function	61
HDF_SD_GETINFO procedure	51	LAGUERRE() function	56	MAGICK_ROWS() function	61
HDF_SD_NAME_TOINDEX() function	51	LAST_ITEM() function	56	MAGICK_WRITE procedure	61
HDF_SD_SELECT() function	51	LA_TRIRED procedure	56	MAGICK_WRITECOLORTABLE procedure	61
HDF_SD_START() function	51	LEGENDRE() function	57	MAGICK_WRITEFILE procedure	61
HDF_VD_ATTACH() function	51	LINDGEN() function	57	MAGICK_WRITEINDEXES procedure	61
HDF_VD_DETACH procedure	51	LINKIMAGE procedure	57	MAKE_ARRAY() function	61
HDF_VD_FIND() function	51	LL_ARC_DISTANCE() function	57	MAP_CLIP_SET procedure	61
HDF_VD_GET procedure	51	LMGR() function	57	MAP_CONTINENTS procedure	61
HDF_VD_READ() function	51	LNGAMMA() function	57	MATRIX_MULTIPLY() function	61
HDF_VG_ATTACH() function	51	LOADCT procedure	57	MAX() function	62
HDF_VG_DETACH procedure	51	LOADCT_INTERNALGDL procedure	58	MEAN() function	62
HDF_VG_GETID() function	51	LOCALE_GET() function	58	MEANABSDEV() function	62
HDF_VG_GETINFO procedure	51	LOGICAL_AND() function	59	MEDIAN() function	62
HDF_VG_GETTRS procedure	52	LOGICAL_OR() function	59	MEMORY() function	62
HEAP_GC procedure	52	LOGICAL_TRUE() function	59	MESSAGE procedure	62
HELP procedure	52	LON64ARR() function	59	MIN() function	62
HELPFORM() function	52	LONARR() function	59	MOMENT() function	62
HISTOGRAM() function	52	LONG() function	59	NCDF_ATT_COPY() function	62
HIST_2D() function	52	LONG64() function	59	NCDF_ATTDEL procedure	62
HIST_ND() function	52	LUDC procedure	59	NCDF_ATTGET procedure	62
IDENTITY() function	53	LUSOL() function	59	NCDF_ATTINQ() function	62
IDL_BASE64() function	53	MACHAR() function	59	NCDF_ATTNAME() function	62
IDL_VALIDNAME() function	53	MAGICK_ADDNOISE procedure	59	NCDF_ATTPUT procedure	62

NCDF_ATTRENAME procedure	63	POLY() function	67	READ_JPEG procedure	71
NCDF_CLOSE procedure	63	POLYFILL procedure	67	READ_PICT procedure	71
NCDF_CONTROL procedure	63	POLY_2D() function	68	READ_PNG() function	71
NCDF_CREATE() function	63	POLY_AREA() function	68	READ_TIFF() function	71
NCDF_DIMDEF() function	63	POPD procedure	68	READ_XWD() function	71
NCDF_DIMID() function	63	PREWITT() function	68	REAL_PART() function	71
NCDF_DIMINQ procedure	63	PRIMES() function	68	REBIN() function	71
NCDF_DIMRENAME procedure	63	PRINT procedure	68	RECALL_COMMANDS() function	71
NCDF_EXISTS() function	63	PRINTD procedure	68	REFORM() function	72
NCDF_INQUIRE() function	63	PRINTF procedure	68	REPLICATE() function	72
NCDF_OPEN() function	63	PRODUCT() function	68	REPLICATE_INPLACE procedure	72
NCDF_VARDEF() function	63	PTRARR() function	68	RESOLVE_ROUTINE procedure	72
NCDF_VARGET procedure	63	PTR_FREE procedure	68	RESTORE procedure	72
NCDF_VARGET1 procedure	64	PTR_NEW() function	69	RETALL procedure	72
NCDF_VARID() function	64	PTR_VALID() function	69	REVERSE() function	72
NCDF_VARINQ() function	64	PUSHD procedure	69	RK4() function	72
NCDF_VARPUT procedure	64	PYTHON procedure	69	RK4JMG() function	72
NCDF_VARRENAME procedure	64	PYTHON() function	69	ROBERTS() function	72
NEWTON() function	64	PY_PLOT procedure	69	ROTATE() function	72
NORM() function	64	PY_PRINT procedure	69	ROUND() function	72
N_ELEMENTS() function	64	QUERY_BMP() function	69	ROUTINE_INFO() function	72
N_PARAMS() function	64	QUERY_DICOM() function	69	ROUTINE_NAMES() function	73
N_TAGS() function	64	QUERY_GIF() function	70	RSTRPOS() function	74
OBJARR() function	64	QUERY_IMAGE() function	70	SAVE procedure	75
OBJ_CLASS() function	64	QUERY_JPEG() function	70	SCOPE_VARFETCH() function	75
OBJ_DESTROY procedure	65	QUERY_PICT() function	70	SEM_CREATE() function	75
OBJ_ISA() function	65	QUERY_PNG() function	70	SEM_DELETE procedure	75
OBJ_NEW() function	65	QUERY_PPM() function	70	SEM_LOCK() function	75
OBJ_VALID() function	66	QUERY_TIFF() function	70	SEM_RELEASE procedure	75
ON_ERROR procedure	66	RADON() function	70	SETENV procedure	75
OPENR procedure	66	RANDOMN() function	70	SET_PLOT procedure	75
OPENU procedure	66	RANDOMU() function	70	SHIFT() function	75
OPENW procedure	66	READ procedure	70	SHOWFONT procedure	75
OPLOT procedure	66	READF procedure	70	SIN() function	83
PARSE_URL() function	66	READS procedure	70	SINDGEN() function	83
PATH_SEP() function	67	READU procedure	70	SINH() function	83
PLOT procedure	67	READ_ASCII() function	71	SIZE() function	84
PLOTERR procedure	67	READ_BINARY() function	71	SKEWNESS() function	84
PLOTS procedure	67	READ_BMP() function	71	SKIP_LUN procedure	84
PM procedure	67	READ_DICOM() function	71	SMOOTH() function	84
POINT_LUN procedure	67	READ_GIF procedure	71	SOBEL() function	84

SOCKET procedure	84	TAN() function	88	WIDGET_DROPLIST() function	91
SORT() function	84	TANH() function	88	WIDGET_EVENT() function	91
SPAWN procedure	84	TEMPLATE procedure	88	WIDGET_INFO() function	91
SPHER_HARM() function	84	TEMPLATE_BLANK procedure	89	WIDGET_LABEL() function	92
SPL_INIT() function	84	TEMPORARY() function	89	WIDGET_TEXT() function	92
SPL_INIT_OLD() function	84	TEST procedure	89	WINDOW procedure	92
SPL_INTERP() function	84	TOTAL() function	89	WRITEU procedure	92
SPL_INTERP_OLD() function	84	TRACE() function	89	WRITE_BMP procedure	92
SQRT() function	85	TRANPOSE() function	89	WRITE_GIF procedure	92
STDDEV() function	85	TRIGRID() function	89	WRITE_JPEG procedure	92
STOP procedure	85	TV procedure	89	WRITE_PICT procedure	92
STRARR() function	85	TVLCT procedure	89	WRITE_PNG procedure	92
STRCMP() function	85	TVRD() function	89	WSET procedure	92
STRCOMPRESS() function	85	TVSCL procedure	89	WSHOW procedure	92
STREGEX() function	85	T_PDF() function	89	WTN() function	93
STRING() function	85	UINDGEN() function	89	XYOUTS procedure	93
STRJOIN() function	85	UINT() function	90		
STRLEN() function	86	UINTARR() function	90	II. Developer's guide	
STRLOWCASE() function	86	UL64INDGEN() function	90	Chapter 16. General remarks and coding guidelines	95
STRMATCH() function	86	ULINDGEN() function	90	Chapter 17. The library-routine API	96
STRMID() function	86	ULON64ARR() function	90	Chapter 18. Extending the documentation	97
STRPOS() function	86	ULONARR() function	90	Chapter 19. Extending the testsuite	
STRPUT procedure	86	ULONG() function	90	(testsuite/README)	98
STRSPLIT() function	86	ULONG64() function	90	Chapter 20. A short overview of how GDL works	
STRTOK() function	86	UNIQ() function	90	internally	99
STRTRIM() function	87	USERSYM procedure	90	Chapter 21. How to make use of OpenMP in GDL	100
STRUCT_ASSIGN procedure	87	VALUE_LOCATE() function	90	Chapter 22. Notes for packagers	101
STRUPCASE() function	87	VARIANCE() function	90	Optional features of PLplot and ImageMagick	101
STR_SEP() function	87	VOIGT() function	90	The HDF4-netCDF conflict	101
SURFACE procedure	87	WAIT procedure	90	III. Indices	
SVDC procedure	88	WDELETE procedure	91	Subject Index	104
SWAP_ENDIAN() function	88	WHERE() function	91	Bibliography	105
SWAP_ENDIAN_INPLACE procedure	88	WIDGET_BASE() function	91		
SYSIME() function	88	WIDGET_BUTTON() function	91		
TAG_NAMES() function	88	WIDGET_CONTROL procedure	91		

About GDL

GNU Data Language (GDL) is a free/libre/open source incremental compiler compatible with IDL and to some extent with PV-WAVE. Together with its library routines it serves as a tool for data analysis and visualization in such disciplines as astronomy, geosciences and medical imaging.

GDL as a language is dynamically-typed, vectorized and has object-oriented programming capabilities. GDL library routines handle numerical calculations, data visualisation, signal/image processing, interaction with host OS and data input/output. GDL supports several data formats such as netCDF, HDF4, HDF5, GRIB, PNG, TIFF, DICOM, etc. Graphical output is handled by X11, PostScript, SVG or z-buffer terminals, the last one allowing output graphics (plots) to be saved in a variety of raster graphics formats. GDL features integrated debugging facilities. GDL has also a Python bridge (Python code can be called from GDL; GDL can be compiled as a Python module).

Packaged versions of GDL are available for several Linux and BSD flavours as well as Mac OS X. The source code compiles as well on other UNIX systems, including Solaris. GDL source code is available for download from Sourceforge.net at: <http://sourceforge.net/projects/gnudatalanguage/>.

Other open-source numerical data analysis tools similar to GDL include:

- GNU Octave: <http://www.gnu.org/software/octave/>
- NCL – NCAR Command Language: <http://www.ncl.ucar.edu/>
- PDL – Perl Data Language: <http://pdl.perl.org/>
- R: <http://www.r-project.org/>
- Scilab: <http://www.scilab.org/>
- SciPy: <http://www.scipy.org/>
- Yorick: <http://yorick.sourceforge.net/>

License

GDL is a free, libre and open-source software released under the GNU General Public License version 2 Foundation [1]. It basically means that any GDL user has the freedom to run, copy, distribute, study, change and improve GDL.

Credits

GDL have been developed by a team of volunteers led by **Marc Schellens** – the project's founder and maintainer. As of 2011 the core team consists additionally of (in alphabetical order) Sylwester Arabas, Alain Coulais and Jeol Gales.

Among many good folks who provided patches and valuable feedback (in alphabetical order) there are: Médéric Bocquien, Justin Bronn, Pierre Chaniel, Pedro Corona Romero, Gilles Duvert, Christoph Fuchs, Nicolas Galmiche, Greg Huey, Gaurav Khanna, Christopher Lee, Maxime Lenoir, Peter Messmer, Gregory Marchal, Thibaut Mermet, Lea Noreskal, Orion Poplawski, Rene Preusker, Mateusz Turcza, Joanna Woo, H Xu, ...

GDL contains snippets of code borrowed from other free and open-source projects credited to: Deepak Bandyopadhyay, Sergio Gelato, Lutz Kettner, Craig B. Markwardt, Paul Ricchiazzi, Danny Smith, J.D. Smith, Richard Schwartz, Paul Wessel, Bob Withers, ...

Pre-compiled or pre-configured packages of GDL are available for numerous systems thanks to: Juan A. Añel, Axel Beckert, Markus Dittrich, Takeshi Enomoto, Sébastien Fabbro, Orlando Garcia Feal, Gaurav Khanna, Justin Lecher, Sebastien Maret, Lea Noreskal, Orion Poplawski, Marius Schamschula, Gürkan Sengün, Thierry Thomas, ...

GDL is written in C++ using the Terence Parr's ANTLR language-recognition framework. Most of the library routines are implemented as interfaces to open-source packages such as GNU Scientific Library, PLPlot, FFTW, ImageMagick, and many many more.

Last but not least, we would like to acknowledge the designers of IDL and PV-WAVE.

Please do report any missing name on the lists above in the same way as any other bug in GDL (see section below).

Providing feedback

Your comments are welcome! Let us know what you use GDL for. Or if you don't, why not. Which functionality are you missing/would appreciate most for coming versions. Please send your bug reports, complaints, suggestions, comments and patches using the trackers or forums available at GDL's project website at SourceForge: <http://sourceforge.net/projects/gnudatalanguage/>.

Organization of this document

This document is divided into two parts:

- User's guide: intended for users developing programs written in GDL,
- Developer's guide: intended for those interested in developing or packaging GDL.

Most of GDL functionalities are exemplified with short GDL scripts. For each such script there are two listings provided: a source code listing with line numbers to the left and a log of output below, e.g.:

```
1 print , 'Hello world!'
```

```
Hello world!
```

All scripts are run by invoking `gdl script.pro` what is equivalent to loading the script with the `@` operator or typing every line of script at the GDL's interactive mode command prompt.

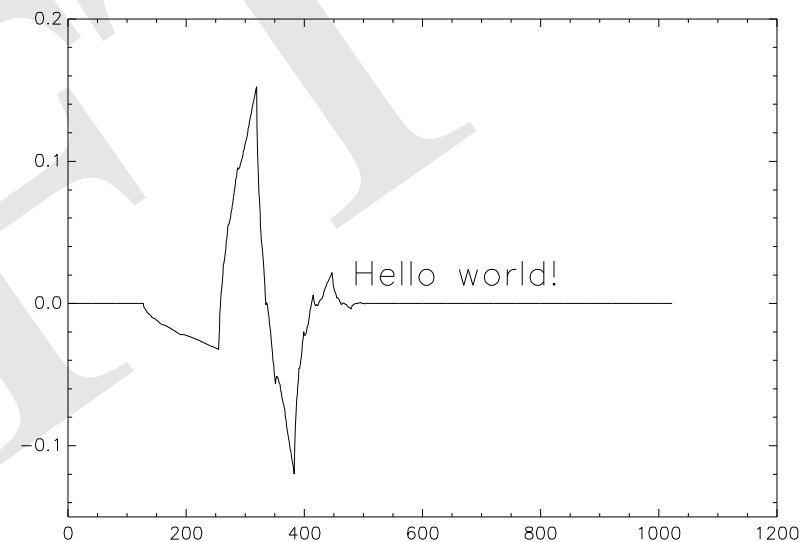
Often the scripts contain lines beginning with a dollar sign "\$" which is the GDL syntax for executing shell commands, e.g.

```
1 $ echo "Hello world!"
```

```
Hello world!
```

If a script involves creation of a plot, the resultant postscript file is displayed below the output listing, e.g.:

```
1 plot , wtn([fltarr(9), 1, fltarr(1014)], 4, /inverse)
2 xyouts , 480, .02, 'Hello world!', charsize=2
```



While GDL itself reached a beta status of development, the hereby documentation is far from reaching an alpha status – **help is very welcome!**

Part I

User's guide

Chapter 1

Obtaining, installing, and invoking GDL

Requirements and supported environments

Availability of pre-compiled packages

Compiling GDL from source

Compiler requirements

GNU g++ clang Intel C++

Autotools

Cmake

Installation layout

Command-line options

Influential environmental variables

Chapter 2

Language reference

Syntax basics

IDL_VALIDNAME() TEMPORARY()

Datatypes

ASSOC()

BYTE() COMPLEX(), DCOMPLEX() (CONJ(), ATAN(), IMAGINARY(), REAL_PART())

DOUBLE() FIX() FLOAT() LONG() LONG64() UINT() ULONG() ULONG64()

SIZE()

Operators

LOGICAL_AND() LOGICAL_OR() LOGICAL_TRUE()

SQRT()

Flow control structures

Conditional execution

IF

```
1 a = 10
2 if a gt 5 then print, 'a is greater than 5'
```

```
a is greater than 5
```

```
1 a = 10
2 if a gt 5 then print, 'a > 5' else print, 'a <= 5'
```

```
a > 5
```

contrary to... cannot be used in interactive mode nor in batch scripts, but only within ...

```
1 $ cat replace_with_nans.pro
2 x = [1.1, 2.1, -3.3, 4.1, -999, 6]
3 replace_with_nans, x, -999
4 print, x
```

```
pro replace_with_nans, x, val
  whr = where(x eq val, cnt)
  if cnt gt 0 then begin
    x[whr] = !VALUES.F_NAN
    message, 'nan count: ' + strtrim(cnt, 2), /cont
  endif
end
% Compiled module: REPLACE_WITH_NANS.
% REPLACE_WITH_NANS: nan count: 1
      1.10000      2.10000      -3.30000      4.10000
```

nan

data type	size	constants	min	max	casting	array allocation	index array alloc.	freeing
natural numbers incl. zero (unsigned)	8b	1b	0	255	BYTE()	BYTARR()	BINDGEN()	TEMPORARY()
	16b	1u	0	65535	UINT()	UINTARR()	UINDGEN()	
	32b	1ul	0	$4 \cdot 10^9$	ULONG()	ULONARR()	ULINDGEN()	
	64b	1ull	0	$1,8 \cdot 10^{19}$	ULONG64()	ULON64ARR()	UL64INDGEN()	
integer numbers (signed)	16b	1	-32768	32767	FIX()	INTARR()	INDGEN()	TEMPORARY()
	32b	1l	$-2 \cdot 10^9$	$2 \cdot 10^9$	LONG()	LONARR()	LINDGEN()	
	64b	1ll	$-9 \cdot 10^{18}$	$9 \cdot 10^{18}$	LONG64()	LONG64ARR()	L64INDGEN()	
real numbers	32b	1.	-10^{38}	10^{38}	FLOAT()	FLTARR()	FINDGEN()	TEMPORARY()
	64b	1d	-10^{308}	10^{308}	DOUBLE()	DBLARR()	DINDGEN()	
complex numbers	64b	complex(1,0)	2x float	2x float	COMPLEX()	COMPLEXARR()	CINDGEN()	TEMPORARY()
	128b	dcomplex(1,0)	2x double	2x double	DCOMPLEX()	DCOMPLEXARR()	DCINDGEN()	
character (byte) strings	variable	'one'	–	–	STRING()	STRARR()	–	TEMPORARY()
structures	variable	{a:1, b:1}	–	–	–	REPLICATE()	–	TEMPORARY()
pointers	n/a	ptr_new(1)	–	–	–	PTRARR()	–	PTR_FREE()
objects	n/a	obj_new('One')	–	–	–	OBJARR()	–	OBJ_DESTROY()

CASE

SWITCH

Loops

FOR

FOREACH

FOREACH statement allows to simplify loop constructs when the array index is not used within the loop:

```
1 tocompare = ['apples', 'oranges']
2 foreach a, tocompare do help, a
```

```
A          STRING    = 'apples'
A          STRING    = 'oranges'
```

As with index variables in FOR loops, the lifetime of the "loop variables" in FOREACH statements extends beyond the loop execution (see example below). Both BREAK and CONTINUE statements work in FOREACH in the same way as in other loop constructs:

```
1 $ cat example.pro
2 example
```

```
pro example
  letters = ['a', 'b', 'c', 'd', 'e']
  foreach l, letters do begin
    if l eq 'c' then continue
    if l eq 'd' then break
    print, 'trying to replace '+ l + ' with 'x''
    l = 'x'
  endforeach
  print, letters
  print, l
end
% Warning: Assignment to loop variable detected.
```

```
% Compiled module: EXAMPLE.
trying to replace a with 'x'
trying to replace b with 'x'
a b c d e
d
```

Loop variables in FOREACH statements contain copies of the array elements thus assigning them a value within the loop does not change contents of the array and as a potentially bug-prone situation causes a compiler warning (see example above).

REPEAT

WHILE

Jumps

GOTO

Highly deprecated as it usually make the code difficult to read and prone to errors. Anyhow, the syntax is as follows

```
1 $ cat example.pro
2 example

pro example
  x = 0
  goto , a
  x++
  a: print , 'x = ', x
end
% Compiled module: EXAMPLE.
x = 0
```

As most of the flow control operator described in this section GOTO is usable only within a GDL routine – not within a batch script which is equivalent to a series of statements in the interactive mode.

Other

EXECUTE()

Variable scoping rules

Functions and procedures

There may exist a function and a procedure of the same name (e.g. PYTHON() and PYTHON, CALL_METHOD() and CALL_METHOD)

EXPAND_PATH(), FILEPATH()
CALL_FUNCTION() CALL_PROCEDURE()

Argument passing

N_PARAMS() KEYWORD_SET() ARG_PRESENT() N_ELEMENTS() SIZE()
_EXTRA _STRICT_EXTRA _REF_EXTRA

when by reference, when by value...

Keyword name abbreviations are allowed if unambiguous, e.g.:

```
1 help , strpos('kayak' , 'a' , /reverse_search)
2 help , strpos('kayak' , 'a' , /reverse_s)
3 help , strpos('kayak' , 'a' , /rev)
4 help , strpos('kayak' , 'a' , 2, /reverse_search , /reverse_offset)

<Expression>    LONG    =    3
<Expression>    LONG    =    3
% STRPOS: Ambiguous keyword abbreviation: REV
% Execution halted at: $MAIN$
<Expression>    LONG    =    1
```

Arrays

```
PRINT ( TV) PM
N_ELEMENTS() SIZE()
REFORM() REBIN() REVERSE() ROTATE() TRANSPOSE()
SORT() UNIQ()
WHERE() ARRAY_INDICES()
ARRAY_EQUAL()
MAKE_ARRAY() REPLICATE() REPLICATE_INPLACE
BYTARR() COMPLEXARR() DBLARR() DCOMPLEXARR() FLTARR() INTARR()
LON64ARR() LONARR() OBJARR() PTRARR() STRARR() UINTARR()
ULON64ARR() ULONARR()
BINDGEN() CINDGEN() DCINDGEN() DINDGEN() FINDGEN() INDGEN()
L64INDEGEN() LINDEGEN() SINDGEN() UINDGEN() UL64INDGEN() ULINDGEN()
IDENTITY()
```

Structures

```
CREATE_STRUCT() N_TAGS() STRUCT_ASSIGN TAG_NAMES()
```

System variables (global)

```
DEFSYSV (checking if running GDL)
```

Heap variables (pointers)

```
HEAP_GC PTRARR PTR_FREE PTR_NEW() PTR_VALID()
```

The HELP procedure

```
HELP
```

Object-oriented programming

```
CALL_METHON CALL_METHON() OBJARR()
OBJ_CLASS() OBJ_DESTROY OBJ_ISA() OBJ_NEW() OBJ_VALID()
```

Handling Overflows, Floating Point Special Values

```
CHECK_MATH() FINITE() MACHAR()
```

Error handling

```
MESSAGE CATCH ON_ERROR ON_IOERROR EXECUTE
```

Compile options

```
1 $ cat example.pro
2 help, 1
3 example
```

```
pro example
  compile_opt idl2
  help, 1
end
<Expression>    INT      =      1
% Compiled module: EXAMPLE.
<Expression>    LONG     =      1
```

```
1 $cat example.pro
2 example
```

```
pro example_helper
  compile_opt hidden
  print, 'example procedure helper'
end
pro example
```

```
    example_helper  
end  
% Compiled module: EXAMPLE.  
example procedure helper
```


Chapter 3

Interpreter commands and built-in debugging facilities

MESSAGE RETALL STOP .COMPILE .STEP .CONTINUE

CHECK_MATH

JOURNAL RECALL_COMMANDS

MEMORY (TEMPORARY())

RESOLVE_ROUTINE ROUTINE_INFO() ROUTINE_NAMES() SCOPE_VARFETCH()

Chapter 4

Maths

Basic Scalar, vector and array operations

TOTAL() SQRT() REVERSE() SHIFT() MAX() MIN() MEAN() NORM()
CONVOL() PRODUCT() CROSSP() DERIV() INVERT() MATRIX_MULTIPLY()
TRACE() TRANSPOSE() (ROTATE())
UNIQ()?

Basic and special function library

GDL has a built-in collection of mathematical functions that are listed below. A great majority of these routines accept both scalar and vector arguments of any numerical type and return the result as scalars or vectors, respectively, preserving the type of the argument, e.g.:

```
1 help , abs(-1) , abs([-!PI,0,!PI])
```

<Expression>	LONG	=	1
<Expression>	FLOAT	=	Array[3]

Some of the routines support a /DOUBLE keyword (flag) which enables one to force GDL to perform the calculations in (if applicable) and return the value[s] as double precision floating point numbers regardless of the type of the argument[s] passed, e.g:

```
1 help , gamma(36b) , gamma(36b , /double)
```

<Expression>	FLOAT	=	inf
<Expression>	DOUBLE	=	1.0333148e+40

Similarly, if a functions returns integer numbers, the /L64 keyword (flag) can be used to force usage of 64-bit integers, e.g.:

```
1 help , round(1d10) , round(1d10 , /l64)
```

<Expression>	LONG	=	-2147483648
<Expression>	LONG64	=	10000000000

If GDL was compiled with OpenMP support (which is the default if the compiler supports it, and most of them do nowadays), and if GDL is run on a multi-cpu (or multi-core) system, and if the array[s] passed as the argument[s] are big enough (see chapter ... TODO) the computations are performed by multiple threads. Consult the individual documentation entries of each of the routines for details.

ABS() returns the absolute value[s] of the real number[s] passed as the argument (integer or floating point) or the magnitude[s] in case of complex number[s]

CEIL() returns the smallest integer number[s] greater than or equal to the argument

FLOOR() returns the greatest integer number[s] less than or equal to the argument (aka the Gauss' symbol)

ROUND() returns an integer value[s] closest to the argument

ERF()

IMSL_ERF()

ERFC()

ERRORF()

EXPINT()

ALOG()

ALOG10()

EXP() (**GSL_EXP()**)

... the following trigonometric functions:

SIN() returns the sine of the argument
ASIN() returns the cosine of the argument
COS()
ACOS()
TAN()
ATAN() ... complex! ...

the following hyperbolic functions:

SINH()
COSH()
TANH()

as well as the following related functions:

LL_ARC_DISTANCE()

BESELI()
BESELJ()
BESELK()
BESELY()

SPHER_HARM()
LAGUERRE()
LEGENDRE()

GAUSSINT() **GAUSS_CVF()** **GAUSS_PDF()**
T_PDF()
FACTORIAL() **GAMMA()** **BETA()** **IGAMMA()** **LNGAMMA()**
PRIMES()
VOIGT()

Linear algebra

LA_TRIRED **LUDC** **SVDC**
IDENTITY() **REPLICATE()** **REPLICATE_INPLACE**

Statistics

CORRELATE()
HISTOGRAM() **HIST_2D()** (implemented using **HIST_ND()**)
IMSL_BINOMIALCOEF()
GAUSSINT() **GAUSS_CVF()** **GAUSS_PDF()**
T_PDF()
KURTOSIS() **SKEWNESS()** **MEAN()** **MIN()** **MAX()** **MEDIAN()** **MEANABS-DEV()** **MOMENT()** **STDDEV()** **VARIANCE()**

Interpolation

INTERPOL() (implemented using **FINDEX()**) **INTERPOLATE()**
REBIN()
DERIV()
SPL_INIT() **SPL_INTERP()**
VALUE_LOCATE()

Polynomials

IMSL_ZEROPOLY() **POLY()**

Geometric calculations

POLY_AREA() **TRIGRID()**

Bitwise operations

ISHFT() **BYTEORDER** **SWAP_ENDIAN()** **SWAP_ENDIAN_INPLACE**

Function fitting

Markwardt [3]

Fourier analysis

FFT() DIST()

Multidimensional root-finding

BROYDEN() IMSL_ZEROPOLY() NEWTON()

Random numbers

RANDOMN() RANDOMU()

Ordinary differential equations

RK4()

Wavelet analysis

WTN()

Mathematical and physical constants

!PI !DPI IDL_CONSTANT()

Chapter 5

Input/output, supported data formats

Basics – accessing files and io streams

```
PRINT PM GET_KBRD READ
BYTEORDER CLOSE EOF
READ WRITE
READF READS READU
GET_LUN FREE_LUN POINT_LUN SKIP_LUN
OPENR OPENU OPENW
```

ASCII

```
PRINTF READF READ_ASCII
```

CSV

Binary data (raw access)

```
READ_BINARY()
BYTEORDER SWAP_ENDIAN() SWAP_ENDIAN_INPLACE
```

FITS

Astron

netCDF

```
NCDF_ATTCOPY() NCDF_ATTDEL NCDF_ATTGET NCDF_ATTINQ() NCDF_ATTNAME(
NCDF_ATTPUT NCDF_ATTRENAME NCDF_CLOSE NCDF_CONTROL NCDF_CREATE()
NCDF_DIMDEF() NCDF_DIMID() NCDF_DIMINQ NCDF_DIMRENAME NCDF_EXISTS()
NCDF_INQUIRE() NCDF_OPEN() NCDF_VARDEF() NCDF_VARGET NCDF_VARGET1
NCDF_VARID() NCDF_VARINQ() NCDF_VARPUT NCDF_VARRENAME
```

HDF4

```
HDF_CLOSE HDF_OPEN()
HDF_SD_ADDDATA HDF_SD_ATTRFIND() HDF_SD_ATTRINFO HDF_SD_CREATE()
HDF_SD_DIMGET HDF_SD_DIMGETID() HDF_SD_END HDF_SD_ENDACCESS
HDF_SD_FILEINFO HDF_SD_GETDATA HDF_SD_GETINFO HDF_SD_NAMETOINDEX()
HDF_SD_SELECT() HDF_SD_START()
HDF_VD_ATTACH() HDF_VD_DETACH HDF_VD_FIND() HDF_VD_GET
HDF_VD_READ()
HDF_VG_ATTACH() HDF_VG_DETACH HDF_VG_GETID() HDF_VG_GETINFO
HDF_VG_GETTRS
```

HDF5

```
H5A_CLOSE H5A_GET_NAME() H5A_GET_NUM_ATTRS() H5A_GET_SPACE()
H5A_GET_TYPE() H5A_OPEN_IDX() H5A_OPEN_NAME() H5A_READ()
H5D_CLOSE H5D_GET_SPACE() H5D_GET_TYPE() H5D_OPEN() H5D_READ()
H5F_CLOSE H5F_IS_HDF5() H5F_OPEN() H5G_CLOSE H5G_OPEN() H5S_CLOSE
H5S_GET_SIMPLE_EXTENT_DIMS() H5T_CLOSE H5T_GET_SIZE() H5_GET_LIBVERSION
```

raster images (TIFF, PNG, JPEG, ...)

see chapter in Image Processing

DICOM**GRIB**

GRIBAPI_CLONE() GRIBAPI_CLOSE_FILE GRIBAPI_COUNT_IN_FILE() GRIB-
API_GET GRIBAPI_GET_DATA GRIBAPI_GET_SIZE() GRIBAPI_NEW_FROM_FILE()
GRIBAPI_OPEN_FILE() GRIBAPI_RELEASE

IDL save files

RESTORE SAVE

Chapter 6

Plotting and mapping

2D plots

AXIS CONTOUR OPLOT PLOT PLOTERR PLOTS POLYFILL XYOUTS

3D plots

SURFACE PLOTS

Plotting raster data

BYTSCL() TV() TVLCT() TVRD() TVSCL()

Managing multiple windows

WDELETE WINDOW WSHOW WSET

Map projections

MAP_CONTINENTS MAP_PROJ_FORWARD MAP_PROJ_INVERSE
LL_ARC_DISTANCE()
MAP_CLIP_SET

Output terminals

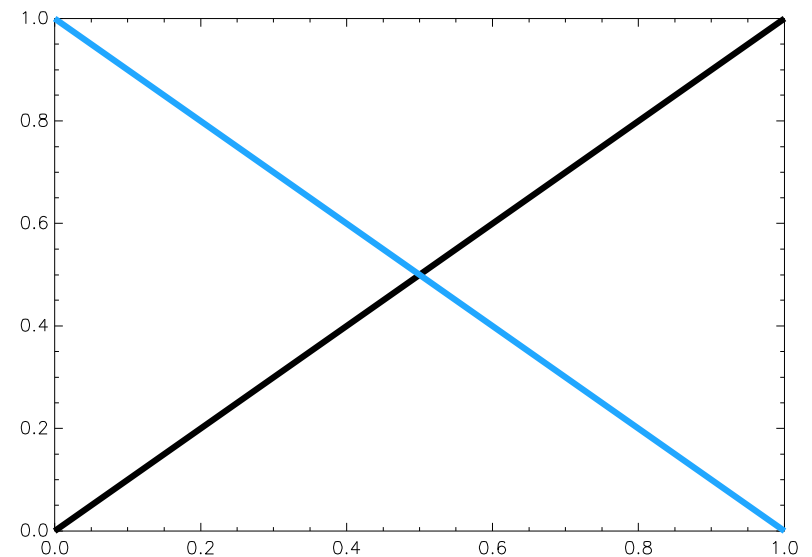
SET_PLOT DEVICE CURSOR ERASE FLUSH

Working with colours

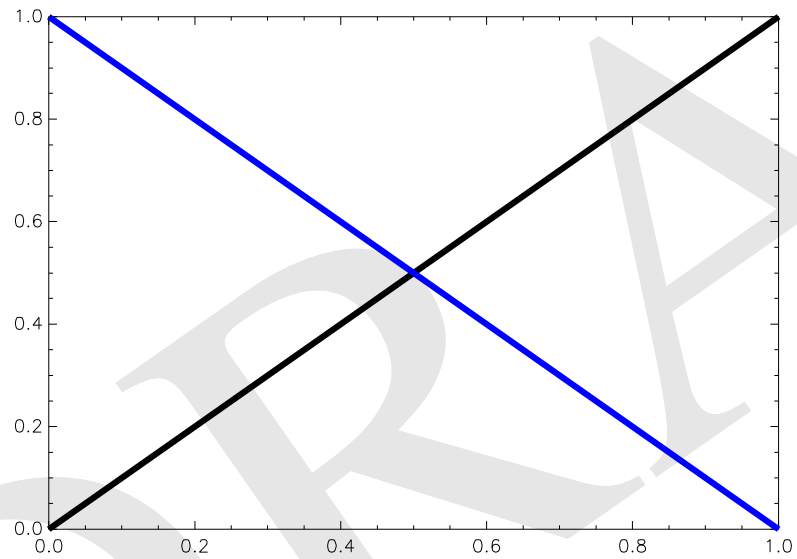
LOADCT

```
1 device , /color , decomposed=0
2 loadct , 1
3 plot , [0,1] , thick=20
4 oplot , [1,0] , color=200 , thick=20
```

```
% Compiled module: LOADCT.
% LOADCT: Loading table BLUE/WHITE
```



```
1 device, /color, decomposed=1  
2 plot, [0,1], thick=20  
3 oplot, [1,0], color='ff0000' x, thick=20
```



Fonts, symbols and text formatting

SHOWFONT Harshey fonts [8]

Misc

CONVERT_COORD() GET_SCREEN_SIZE()

Chapter 7

Interaction with host OS

CD POPD PUSHD PRINTD EXIT WAIT

Executing external commands (via shell or not)

SPAWN (while EXECUTE() ...)

Filesystem operations

CD FILE_BASENAME() FILE_COPY FILE_DELETE FILE_DIRNAME() FILE_EXPAND_PATH()
(EXPAND_PATH()) FILE_INFO() FILE_LINES() FILE_MKDIR FILE_SAME()
FILE_SEARCH() FILE_TEST() FILE_WHICH() FINDFILE() FSTAT() PATH_SEP()

Network operations

SOCKET PARSE_URL()

Command-line options and environmental variables

COMMAND_LINE_ARGS() SETENV GETENV() LOCALE_GET()

Chapter 8

Manipulating strings

STRCMP() STRCOMPRESS() STREGEX() STRJOIN() STRLEN()
STRLOWERCASE() STRUPCASE()
STRMID() STRPOS() RSTRPOS() STRPUT() STRSPLIT() STRTOK() STR-
TRIM() STR_SEP()
READS()
STRARR() STRING() SINDGEN()
IDL_BASE64() IDL_VALIDANEM() SORT() UNIQ() PARSE_URL()

Chapter 9

Representing date & time

CALDAT CALENDAR SYSTIME()

Chapter 10

Image processing

QUERY_BMP() QUERY_DICOM() QUERY_GIF() QUERY_IMAGE() QUERY_JPEG()
QUERY_PICT() QUERY_PNG() QUERY_PPM() QUERY_TIFF()
READ_BMP() READ_DICOM() READ_JPEG READ_PICT READ_PNG() READ_TIFF()
READ_XWD()
WRITE_BMP WRITE_JPEG WRITE_PICT WRITE_PNG
BYTSCL() CONVOL() MEDIAN() POLY_2D() PREWITT() RADON() ROBERTS()
ROTATE() REBIN() SMOOTH() SOBEL()

Chapter 11

Parallel processing

Built-in features (OpenMP)

CPU

Semaphores and shared memory (library routines)

SEM_CREATE() SEM_DELETE SEM_LOCK() SEM_RELEASE

ImageMagick's features

MPI and GDL

Chapter 12

GUI programming (widgets)

DIALOG_MESSAGE() DIALOG_PICKFILE()

WIDGET_BASE() WIDGET_BUTTON() WIDGET_CONTROL WIDGET_DROPLIST()

WIDGET_EVENT() WIDGET_INFO() WIDGET_LABEL() WIDGET_TEXT()

Chapter 13

Dynamic loading

CALL_EXTERNAL() LINKIMAGE()

Chapter 14

The Python bridge

van Rossum and Fred L. Drake [6]

calling Python code from GDL

PYTHON() PYTHON

calling GDL code from Python

Chapter 15

Alphabetical list of library routines

ABS() function

positional arguments: 1

keywords: none

Returns absolute value of a number passed as the first argument or an array of absolute values if argument is an array. For complex arguments the length of the argument in the complex plane is returned (the phase of a complex number may be obtained using **ATAN()**).

```
1 print , abs(-2.2)
2 print , abs([-1,1,0])
3 print , abs(.5 * sqrt(2) * complex(1, 1))
```

```
2.20000
1      1      0
1.00000
```

multi-threading: this routine uses GDL thread pool if working on large array, see the...

ACOS() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

ALOG() function

positional arguments: 1

keywords: none

ALOG10() function

positional arguments: 1

keywords: none

APPLEMAN procedure

positional arguments: 2

keywords: **HELP**, **NODISPLAY**, **RESULT**, **TEST**, **XSIZE**, **YSIZE**

Computes and optionally renders the Mandelbrot set. The two positional arguments are optional and allow specification of the range over which the set is computed (default values: [-1.0,2.3] and [-1.3,1.3]).

RESULT keyword

Allows passing a variable into which the computed data will be stored. If set, no rendering is done.

XSIZE keyword

Allows specification of the width of the domain over which the set is computed.

YSIZE keyword

Allows specification of the height of the domain over which the set is computed.

```

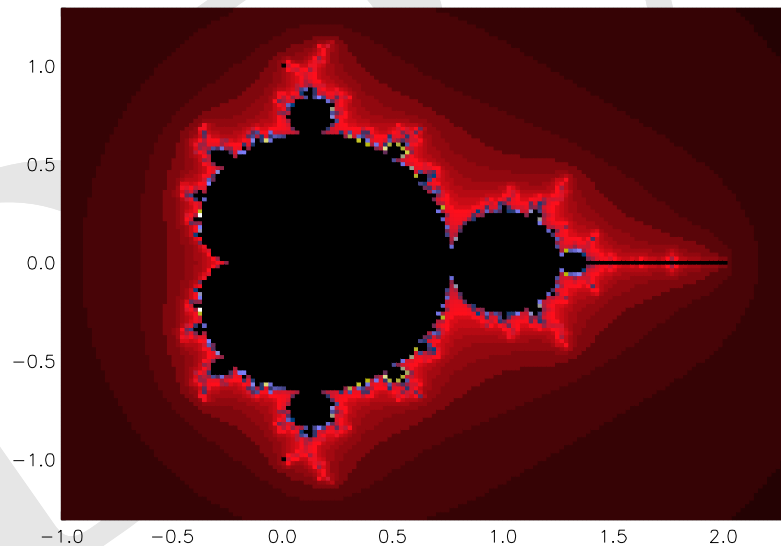
1 rng_x = [-1, 2.3]
2 rng_y = [-1.3, 1.3]
3 appleman, rng_x, rng_y, result=fractal, xsize=165, ysize=130
4 device, /color
5 plot, [0], /nodata, xrange=rng_x, yrange=rng_y
6 loadct, 15
7 tvscl, fractal, rng_x[0], rng_y[0], $
8   xsize=rng_x[1]-rng_x[0], ysize=rng_y[1]-rng_y[0]

```

```

% Compiled module: APPLEMAN.
% Compiled module: LOADCT.
% LOADCT: Loading table BOW SPECIAL
% Compiled module: TVSCL.

```

**ARG_PRESENT() function**

positional arguments: 1

keywords: none

ARRAY_EQUAL() function

positional arguments: 2

keywords: **NO_TYPECONV**

ARRAY_INDICES() function

positional arguments: 2

keywords: none

see also: **WHERE()**

ASIN() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

ASSOC() function

positional arguments: 3

keywords: **PACKED**

ATAN() function

positional arguments: 2

keywords: **PHASE**

multi-threading: this routine uses GDL thread pool if working on large array, see the...

AXIS procedure

positional arguments: 3

keywords: CHARSIZE, CHARTHICK, COLOR, DATA, DEVICE, FONT, NODATA, NOERASE, NORMAL, SAVE, SUBTITLE, T3D, THICK, TICKLEN, XAXIS, XCHARSIZE, XGRIDSTYLE, XLOG, XMARGIN, XMINOR, X RANGE, XSTYLE, XTHICK, XTICKFORMAT, XTICKINTERVAL, XTICKLEN, XTICKNAME, XTICKS, XTITLE, XTYPE, YAXIS, YCHARSIZE, YGRIDSTYLE, YLOG, YMARGIN, YMINOR, YNOZERO, YRANGE, YSTYLE, YTHICK, YTICKFORMAT, YTICKINTERVAL, YTICKLEN, YTICKNAME, YTICKS, YTITLE, YTYPE, ZCHARSIZE, ZGRIDSTYLE, ZMARGIN, ZMINOR, ZRANGE, ZSTYLE, ZTHICK, ZTICKFORMAT, ZTICKLEN, ZTICKNAME, ZTICKS, ZTITLE, ZVALUE

BESELI() function

positional arguments: 2

keywords: DOUBLE, HELP, ITER

BESELJ() function

positional arguments: 2

keywords: DOUBLE, HELP, ITER

BESELK() function

positional arguments: 2

keywords: DOUBLE, HELP, ITER

BESELY() function

positional arguments: 2

keywords: DOUBLE, HELP, ITER

BETA() function

positional arguments: 2

keywords: DOUBLE

BILINEAR() function

positional arguments: 3

keywords: MISSING

BINDGEN() function

positional arguments: 8

keywords: none

BROYDEN() function

positional arguments: 2

keywords: DOUBLE, ITMAX, TOLF, TOLX

BYTARR() function

positional arguments: 8

keywords: NOZERO

BYTE() function

positional arguments: 10

keywords: none

BYTEORDER procedure

positional arguments: any number

keywords: DTOXDR, FTOXDR, HTONL, HTONS, L64SWAP, LSWAP, NTOHL, NTOHS, SSWAP, SWAP_IF_BIG_ENDIAN, SWAP_IF_LITTLE_ENDIAN, XDR-TOD, XDRTOF

BYTSCL() function

positional arguments: 3

keywords: MAX, MIN, NAN, TOP

CALDAT procedure

positional arguments: 7

keywords: none

CALENDAR procedure

positional arguments: 2

keywords: none

An interface to the UNIX *cal* command. Displays a calendar using the current graphics device (i.e. X, PS, ...). The two optional arguments allow to specify a month, or a month and a year.

```
1 calendar, 9, 1983
```

```
% Compiled module: CALENDAR.
```

September 1983						
Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

CALL_EXTERNAL() function

positional arguments: any number

keywords: ALL_GDL, ALL_VALUE, B_VALUE, D_VALUE, F_VALUE, I_VALUE, L64_VALUE, L_VALUE, RETURN_TYPE, STRUCT_ALIGN_BYTES, S_VALUE, UI_VALUE, UL64_VALUE, UL_VALUE, UNLOAD, VALUE

Calls a routine from a sharable object library. The first argument should be a string containing the filename of the sharable object to load (standard library paths are searched). The second argument should be a string with the name of the routine in the image to be called. All subsequent arguments are passed to the routine.

Here is a, hopefully concise, example covering all the steps one could take to write, build and call a C routine from GDL:

```
1 $ echo '$ cat libexample.c'
2 $ cat libexample.c
3 $ echo '$ cat CMakeLists.txt'
4 $ cat CMakeLists.txt
5 $ echo '$ cmake .'
6 $ cmake .|awk '{ print (length($0)>50?substr($0,0,50) "...":$0)}'
7 $ echo
```

```

8 $ echo '$ make'
9 $ make
10 $ echo
11
12 img = 'libexample.' + (!VERSION.OS_NAME eq 'darwin'? "dylib":"so")
13 message, '1d308 vs. a next representable double:', /continue
14 print, format='(E)', 1d308, $
15   call_external(img, 'c_nextafter', 1d308, 2d308, /d_value)
16
17 $ make clean

```

```

$ cat libexample.c
#include <math.h>
double c_nextafter(int argc, void* argv[]) {
    return nextafter(*(double*)argv[0], *(double*)argv[1]);
}

$ cat CMakeLists.txt
project(libexaple C)
cmake_minimum_required(VERSION 2.0)
add_library(example SHARED libexample.c)
set_directory_properties(PROPERTIES ADDITIONAL_MAKE_CLEAN_FILES
    "Makefile;CMakeCache.txt;cmake_install.cmake;CMakeFiles")

$ cmake .
— The C compiler identification is GNU
— Checking whether C compiler has -isysroot
— Checking whether C compiler has -isysroot - yes
— Checking whether C compiler supports OSX deploy...
— Checking whether C compiler supports OSX deploy...
— Check for working C compiler: /usr/bin/gcc
— Check for working C compiler: /usr/bin/gcc — w...
— Detecting C compiler ABI info
— Detecting C compiler ABI info — done
— Configuring done
— Generating done
— Build files have been written to: /Users/slayoo...

$ make
Scanning dependencies of target example
[100%] Building C object CMakeFiles/example.dir/libexample.o

```

```

Linking C shared library libexample.dylib
[100%] Built target example

% $MAIN$: 1d308 vs. a next representable double:
1.0000000000000000E+308
1.0000000000000002E+308

```

RETURN_TYPE keyword

Indicates the type of the return value of the called routine, this value will be returned by CALL_EXTERNAL to GDL. The value of the keyword is interpreted in the same way as the type field of the **SIZE()** function. Possible values for it are those for numeric types except COMPLEX and DCOMPLEX. The default value is 3 (GDL type LONG, which corresponds to C type int). Alternatively one of the following keywords may be used:

B_VALUE keyword

equivalent to RETURN_TYPE=1 (BYTE)

I_VALUE keyword

equivalent to RETURN_TYPE=2 (INTEGER)

L_VALUE keyword

equivalent to RETURN_TYPE=3 (LONG)
This corresponds to the default behaviour.

F_VALUE keyword

equivalent to RETURN_TYPE=4 (FLOAT)

D_VALUE keyword

equivalent to RETURN_TYPE=5 (DOUBLE)

UI_VALUE keyword

equivalent to RETURN_TYPE=12 (UINT)

UL_VALUE keyword

equivalent to RETURN_TYPE=13 (ULONG)

L64_VALUE keyword

equivalent to RETURN_TYPE=14 (LONG64)

UL64_VALUE keyword

equivalent to RETURN_TYPE=15 (ULONG64)

S_VALUE keyword

equivalent to RETURN_TYPE=6 (STRING, the called function should return char*)

ALL_VALUE keyword

The default is to pass all parameters by reference. If this keyword is set, all parameters are passed by value.

UNLOAD keyword

If set (/UNLOAD or UNLOAD=1) the shared object will be unloaded after calling the routine.

STRUCT_ALIGN_BYTES keyword

If set to an integer n, CALL_EXTERNAL assumes that structures in the shared object are aligned at boundaries of n bytes, where n should be a power of 2. If n=0 or if this keyword is not given, the default machine dependent alignment is assumed (normally 4/8 bytes on 32/64 bit systems). It should only be necessary to use this keyword if the called shared object has been compiled with a different alignment, e.g. with #pragma pack(n)

implementation details: This routine uses the dlopen/dlsym/dlclose calls, and thus is available only on systems that support them. It has been tested on Linux, Apple OS X and Solaris.

see also: [LINKIMAGE](#)

disclaimer: CALL_EXTERNAL was implemented in GDL by Christoph Fuchs, who also wrote the documentation for it which was the base for this entry. Copyright: (C) 2010 by Christoph Fuchs. The original file was licensed under GNU GPL v>=2.

CALL_FUNCTION() function

positional arguments: any number

keywords: [_REF_EXTRA](#)

CALL_METHOD procedure

positional arguments: any number

keywords: [_REF_EXTRA](#)

CALL_METHOD() function

positional arguments: any number

keywords: [_REF_EXTRA](#)

CALL_PROCEDURE procedure

positional arguments: any number

keywords: [_REF_EXTRA](#)

CATCH procedure

positional arguments: 1

keywords: [CANCEL](#)

CD procedure

positional arguments: 1

keywords: [CURRENT](#)

CDF_EPOCH procedure

positional arguments: 8

keywords: [BREAKDOWN_EPOCH](#), [COMPUTE_EPOCH](#)

CEIL() function

positional arguments: 1

keywords: L64

multi-threading: this routine uses GDL thread pool if working on large array, see the...

CHECK_MATH() function

positional arguments: 2

keywords: MASK, NOCLEAR, PRINT

CINDGEN() function

positional arguments: 8

keywords: none

CLOSE procedure

positional arguments: any number

keywords: ALL, EXIT_STATUS, FILE, FORCE

COMMAND_LINE_ARGS() function

positional arguments: none

keywords: COUNT

COMPLEX() function

positional arguments: 10

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

COMPLEXARR() function

positional arguments: 8

keywords: NOZERO

CONGRID() function

positional arguments: 4

keywords: CENTER, CUBIC, HELP, INTERP, MINUS_ONE, MISSING, TEST

CONJ() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

CONTOUR procedure

positional arguments: 3

keywords: BACKGROUND, CHARSIZE, CHARTHICK, CLIP, COLOR, C_CHARSIZE, C_COLORS, C_LINestyle, DATA, DEVICE, FILL, FOLLOW, FONT, ISOTROPIC, LEVELS, MAX_VALUE, MIN_VALUE, NLEVELS, NOCLIP, NO-DATA, NOERASE, NORMAL, OVERPLOT, POSITION, SUBTITLE, T3D, THICK, TICKLEN, TITLE, XCHARSIZE, XGRIDSTYLE, XLOG, XMARGIN, XMINOR, XRange, XSTYLE, XTHICK, XTICKFORMAT, XTICKLEN, XTICKNAME, XTICKS, XTICKV, XTICK_GET, XTITLE, XTYPE, YCHARSIZE, YGRIDSTYLE, YLOG, YMARGIN, YMINOR, YRange, YSTYLE, YTHICK, YTICKFORMAT, YTICKLEN, YTICKNAME, YTICKS, YTICKV, YTICK_GET, YTITLE, YTYPE, ZCHARSIZE, ZGRIDSTYLE, ZLOG, ZMARGIN, ZMINOR, ZRange, ZSTYLE, ZTHICK, ZTICKFORMAT, ZTICKLEN, ZTICKNAME, ZTICKS, ZTICKV, ZTICK_GET, ZTITLE, ZTYPE, ZVALUE

CONVERT_COORD() function

positional arguments: 3

keywords: DATA, DEVICE, DOUBLE, NORMAL, T3D, TO_DATA, TO_DEVICE, TO_NORMAL

CONVOL() function

positional arguments: 3

keywords: CENTER, EDGE_TRUNCATE, EDGE_WRAP

CORRELATE() function

positional arguments: 2

keywords: COVARIANCE, DOUBLE

When called with two vector arguments x and y it returns the correlation coefficient r defined as:

$$r = \frac{\text{cov}(x, y)}{\text{stdev}(x) \cdot \text{stdev}(y)} \quad (15.1)$$

where

$$\text{cov}(x, y) = \frac{1}{N-1} \sum_{i=0}^{N-1} (x[i] - \bar{x}) \cdot (y[i] - \bar{y}) \quad (15.2)$$

$$\text{stdev}(x) = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N-1} [x[i] - \bar{x}]^2} \quad (15.3)$$

and

$$\bar{x} = \sum_{i=0}^{N-1} \frac{x[i]}{N} \quad (15.4)$$

(N is the length of the longer vector).

```
1 print, correlate([-1,0,1], [1,0,-1])
```

```
% Compiled module: CORRELATE.
% Compiled module: MEAN.
-1.00000
```

DOUBLE keyword

Forces double-precision calculations and output value type.

```
1 x = [1, 2, 3, 4, 5]
2 y = [1.1, 1.9, 3.1, 3.9, 5, 6, 7, 8, 9]
3 help, correlate(x, y)
4 help, correlate(x, y, /double)
```

```
% Compiled module: CORRELATE.
% Compiled module: MEAN.
<Expression>    FLOAT    =    0.99813
<Expression>    DOUBLE   =    0.9981310
```

COVARIANCE keyword

If called with the COVARIANCE keyword, the covariance $\text{cov}(x, y)$ of the two vectors is returned instead.

```
1 x = [-1, 0, 1.]
2 y = [-2, 0, 2.]
3 print, correlate(x, y, /covariance)
```

```
% Compiled module: CORRELATE.
% Compiled module: MEAN.
2.00000
```

COS() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

COSH() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

CPU procedure

positional arguments: none

keywords: RESET, RESTORE, TPOOL_MAXELTS, TPOOL_MINELTS, TPOOL_NTHREADS, VECTOR_ENABLE

CREATE_STRUCT() function

positional arguments: any number

keywords: NAME

CROSSP() function

positional arguments: 2

keywords: none

CURSOR procedure

positional arguments: 3

keywords: CHANGE, DATA, DEVICE, DOWN, NORMAL, NOWAIT, UP, WAIT

DBLARR() function

positional arguments: 8

keywords: NOZERO

DCINDGEN() function

positional arguments: 8

keywords: none

DCOMPLEX() function

positional arguments: 10

keywords: none

DCOMPLEXARR() function

positional arguments: 8

keywords: NOZERO

DEFSYSV procedure

positional arguments: 3

keywords: EXISTS

DERIV() function

positional arguments: 2

keywords: HELP, NO_CHECK, TEST

DETERM() function

positional arguments: 1

keywords: DOUBLE

DEVICE procedure

positional arguments: none

keywords: CLOSE_FILE, COLOR, DECOMPOSED, ENCAPSULATED, FILE-NAME, GET_DECOMPOSED, GET_SCREEN_SIZE, GET_VISUAL_DEPTH, INCHES, LANDSCAPE, PORTRAIT, SCALE_FACTOR, SET_CHARACTER_SIZE, SET_RESOLUTION, WINDOW_STATE, XOFFSET, XSIZE, YOFFSET, YSIZE, Z_BUFFERING

DIALOG_MESSAGE() function

positional arguments: 1

keywords: CANCEL, CENTER, DEFAULT_CANCEL, DEFAULT_NO, DIALOG_PARENT, DISPLAY_NAME, ERROR, HELP, INFORMATION, QUESTION, RESOURCE_NAME, TITLE, ZENITY_NAME, ZENITY_PATH

DIALOG_PICKFILE() function

positional arguments: none

keywords: DEBUG, DEFAULT_EXTENSION, DIALOG_PARENT, DIRECTORY, DISPLAY_NAME, FILE, FILTER, FIX_FILTER, GET_PATH, GROUP, HELP, MULTIPLE_FILES, MUST_EXIST, OVERWRITE_PROMPT, PATH, READ, RESOURCE_NAME, TEST, TITLE, VERBOSE, WRITE, ZENITY_NAME, ZENITY_PATH, ZENITY_SEP

DINDGEN() function

positional arguments: 8

keywords: none

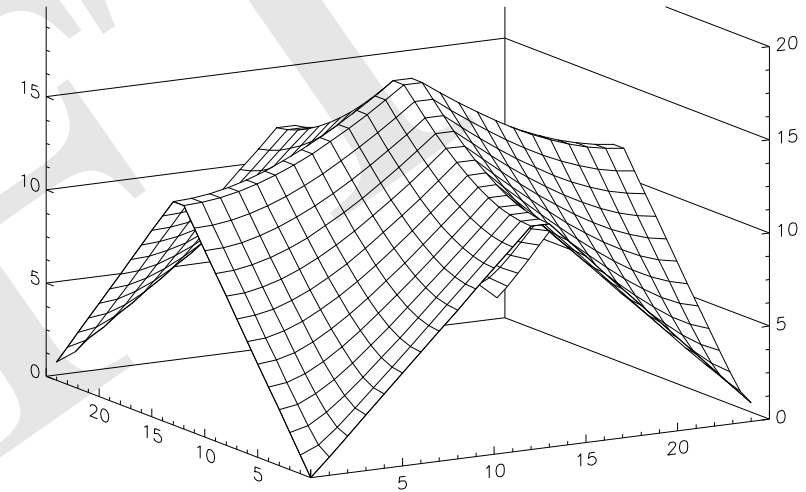
DIST() function

positional arguments: 2

keywords: none

```
1 surface , dist(25)
```

```
% Compiled module: DIST.
```



DOUBLE() function

positional arguments: 10

keywords: none

EOF() function

positional arguments: 1

keywords: none

ERASE procedure

positional arguments: 1

keywords: none

ERF() function

positional arguments: 1

keywords: DOUBLE

ERFC() function

positional arguments: 1

keywords: **DOUBLE**

ERRORF() function

positional arguments: 1

keywords: **DOUBLE**

ESCAPE_SPECIAL_CHAR() function

positional arguments: 1

keywords: **HELP**, **LIST_OF_SPECIAL_CHAR**, **SHOW_LIST**, **TEST**, **VERBOSE**

EXECUTE() function

positional arguments: 2

keywords: none

Executes the statement passed in the first argument, returns 1 if no error occurred or 0 if the execution failed, e.g.

```

1 status = execute('print, "Hello world!"')
2 help, status
3 status = execute('print, Hello world!')
4 help, status

```

```

Hello world!
STATUS      INT      =      1
% Parser syntax error: unexpected token: HELLO
STATUS      INT      =      0

```

EXIT procedure

positional arguments: none

keywords: **NO_CONFIRM**, **STATUS**

STATUS keyword

```

1 spawn, '../ ../../ ../ src/gdl -quiet -e "exit, status=44" 1>/dev/null ', $
2   exit_status=s
3   print, 'spawned GDL process exited with code ', strtrim(s, 2)

```

spawned GDL process exited with code 44

EXP() function

positional arguments: 1

keywords: none

```

1 print, exp([0, 1, -!VALUES.F_INFINITY])
2 print, alog(exp([!PI]))

```

```

1.00000      2.71828      0.00000
3.14159

```

multi-threading: this routine uses GDL thread pool if working on large array, see the...

EXPAND_PATH() function

positional arguments: 1

keywords: **ALL_DIRS**, **ARRAY**, **COUNT**

EXPINT() function

positional arguments: 2

keywords: **DOUBLE**

FACTORIAL() function

positional arguments: 1

keywords: **STIRLING**, **UL64**

FFT() function

positional arguments: 2

keywords: DIMENSION, DOUBLE, INVERSE, OVERWRITE

$$F[m] = \frac{1}{N} \sum_k f[k] \cdot e^{-\frac{2\pi i}{N} mk} \quad (15.5)$$

```
1 $ tail stddev*.pro
2 x = [1.31, 2.44, 2.51, 3.01, 2.96, 2.50, 0.05, 3.24, 0.13]
3 print, stddevsum(x), stddevfft(x)
```

```
=> stddevfft.pro <==
function stddevfft, x
  return, sqrt(total((abs(fft(x))^2)[1:-1]))
end
```

```
=> stddevsum.pro <==
function stddevsum, x
  return, sqrt(mean(x^2) - mean(x)^2)
end
% Compiled module: STDDEVSUM.
% Compiled module: MEAN.
% Compiled module: STDDEVFFT.
1.15258      1.15258
```

multi-threading: this routine uses GDL thread pool if working on large array, see the...

implementation details: FFTW vs. GSL - TODO

FILEPATH() function

positional arguments: 1

keywords: ROOT_DIR, SUBDIRECTORY, TERMINAL, TMP

FILE_BASENAME() function

positional arguments: 2

keywords: FOLD_CASE, HELP

```
1 print, file_basename('/etc/passwd')
```

```
% Compiled module: FILE_BASENAME.
% Compiled module: ESCAPE_SPECIAL_CHAR.
% Compiled module: STRSPLIT.
% Compiled module: UNIQ.
passwd
```

```
1 print, file_basename('/etc/resolv.conf', '.conf')
```

```
% Compiled module: FILE_BASENAME.
% Compiled module: ESCAPE_SPECIAL_CHAR.
% Compiled module: STRSPLIT.
% Compiled module: UNIQ.
resolv
```

```
1 print, file_basename(file_search('../..../src/gdl*.g'))
```

```
% Compiled module: FILE_BASENAME.
% Compiled module: ESCAPE_SPECIAL_CHAR.
% Compiled module: STRSPLIT.
% Compiled module: UNIQ.
gdlc.g gdlc.i.g gdlc.tree.g
```

see also: FILE_DIRNAME(), PATH_SEP()

FILE_COPY procedure

positional arguments: 2

keywords: ALLOW_SAME, HELP, NOEXPAND_PATH, OVERWRITE, QUIET, RECURSIVE, REQUIRE_DIRECTORY, TEST, VERBOSE

FILE_DELETE procedure

positional arguments: 30

keywords: ALLOW_NONEXISTENT, HELP, NOEXPAND_PATH, QUIET, RECURSIVE, TEST, VERBOSE

FILE_DIRNAME() function

positional arguments: 1

keywords: **HELP**, **MARK_DIRECTORY**

FILE_EXPAND_PATH() function

positional arguments: 1

keywords: none

FILE_INFO() function

positional arguments: 2

keywords: **NOEXPAND_PATH**

FILE_LINES() function

positional arguments: 1

keywords: **COMPRESS**, **NOEXPAND_PATH**

```
1 print, file_lines ( '../..../ ChangeLog ')
```

```
% Compiled module: FILE_LINES.  
6335
```

FILE_MKDIR procedure

positional arguments: any number

keywords: **NOEXPAND_PATH**

implementation details: Current implementation uses the system() call and executes the mkdir using using a shell subprocess

FILE_SAME() function

positional arguments: 2

keywords: **NOEXPAND_PATH**

FILE_SEARCH() function

positional arguments: 2

keywords: **COUNT**, **EXPAND_ENVIRONMENT**, **EXPAND_TILDE**, **FOLD_CASE**, **FULLY_QUALIFY_PATH**, **ISSUE_ACCESS_ERROR**, **MARK_DIRECTORY**, **MATCH_ALL_INITIAL_DOT**, **MATCH_INITIAL_DOT**, **NOSORT**, **QUOTE**

FILE_TEST() function

positional arguments: 1

keywords: **BLOCK_SPECIAL**, **CHARACTER_SPECIAL**, **DIRECTORY**, **EXECUTABLE**, **GET_MODE**, **NAMED_PIPE**, **NOEXPAND_PATH**, **READ**, **REGULAR**, **SOCKET**, **SYMLINK**, **WRITE**, **ZERO_LENGTH**

FILE_WHICH() function

positional arguments: 2

keywords: **DEBUG**, **HELP**, **INCLUDE_CURRENT_DIR**, **TEST**

FINDEX() function

positional arguments: 2

keywords: none

FINDFILE() function

positional arguments: 1

keywords: **COUNT**, **HELP**, **QUIET**, **SH_LOCATION**, **SPAWN_OPTIONS**, **TEST**, **VERBOSE**

FINDGEN() function

positional arguments: 8
keywords: none

FINITE() function

positional arguments: 1
keywords: INFINITY, NAN

FIX() function

positional arguments: 10
keywords: PRINT, TYPE

FLOAT() function

positional arguments: 10
keywords: none

FLOOR() function

positional arguments: 1
keywords: L64

multi-threading: this routine uses GDL thread pool if working on large array, see the...

FLTARR() function

positional arguments: 8
keywords: NOZERO

FLUSH procedure

positional arguments: any number
keywords: none

FREE_LUN procedure

positional arguments: any number
keywords: EXIT_STATUS, FORCE

FSTAT() function

positional arguments: 1
keywords: none

GAMMA() function

positional arguments: 1
keywords: DOUBLE

GAUSSINT() function

positional arguments: 1
keywords: DOUBLE

GAUSS_CVF() function

positional arguments: 1
keywords: none

GAUSS_PDF() function

positional arguments: 1
keywords: none

GDL_ERFINV() function

positional arguments: 1
 keywords: **DOUBLE**

GETENV() function

positional arguments: 1
 keywords: **ENVIRONMENT**

GET_DRIVE_LIST() function

positional arguments: none
 keywords: **COUNT**

GET_KBRD() function

positional arguments: 1
 keywords: none

GET_LOGIN_INFO() function

positional arguments: none
 keywords: none

Returns a structure with current username and hostname:

```
1 help, get_login_info(), /structure
```

```
** Structure <Anonymous>, 2 tags, data length=16:
  MACHINE_NAME  STRING  'eyrie.prac.igf'
  USER_NAME     STRING  'slayoo'
```

GET_LUN procedure

positional arguments: 1
 keywords: none

GET_SCREEN_SIZE() function

positional arguments: 1
 keywords: **RESOLUTION**

GRIBAPI_CLONE() function

positional arguments: 1
 keywords: none

GRIBAPI_CLOSE_FILE procedure

positional arguments: 1
 keywords: none

GRIBAPI_COUNT_IN_FILE() function

positional arguments: 1
 keywords: none

GRIBAPI_GET procedure

positional arguments: 3
 keywords: none

GRIBAPI_GET_DATA procedure

positional arguments: 4
 keywords: none

GRIBAPI_GET_SIZE() function

positional arguments: 2

keywords: none

GRIBAPI_NEW_FROM_FILE() function

positional arguments: 1

keywords: none

GRIBAPI_OPEN_FILE() function

positional arguments: 1

keywords: none

GRIBAPI_RELEASE procedure

positional arguments: 1

keywords: none

GSL_EXP() function

positional arguments: 1

keywords: none

H5A_CLOSE procedure

positional arguments: 1

keywords: none

H5A_GET_NAME() function

positional arguments: 1

keywords: none

H5A_GET_NUM_ATTRS() function

positional arguments: 1

keywords: none

H5A_GET_SPACE() function

positional arguments: 1

keywords: none

H5A_GET_TYPE() function

positional arguments: 1

keywords: none

H5A_OPEN_IDX() function

positional arguments: 2

keywords: none

H5A_OPEN_NAME() function

positional arguments: 2

keywords: none

H5A_READ() function

positional arguments: 1

keywords: none

H5D_CLOSE procedure

positional arguments: 1

keywords: none

H5D_GET_SPACE() function

positional arguments: 1

keywords: none

H5D_GET_TYPE() function

positional arguments: 1

keywords: none

H5D_OPEN() function

positional arguments: 2

keywords: none

H5D_READ() function

positional arguments: 1

keywords: none

H5F_CLOSE procedure

positional arguments: 1

keywords: none

H5F_IS_HDF5() function

positional arguments: 1

keywords: none

H5F_OPEN() function

positional arguments: 1

keywords: none

H5G_CLOSE procedure

positional arguments: 1

keywords: none

H5G_OPEN() function

positional arguments: 2

keywords: none

H5S_CLOSE procedure

positional arguments: 1

keywords: none

H5S_GET_SIMPLE_EXTENT_DIMS() function

positional arguments: 1

keywords: none

H5T_CLOSE procedure

positional arguments: 1

keywords: none

H5T_GET_SIZE() function

positional arguments: 1

keywords: none

H5_GET_LIBVERSION() function

positional arguments: none

keywords: none

Returns a string containing the version number of the HDF5 library.

```
1 help , h5_get_libversion ()
```

```
<Expression>    STRING    = '1.8.8'
```

HDF_CLOSE procedure

positional arguments: 1

keywords: none

HDF_OPEN() function

positional arguments: 2

keywords: ALL, CREATE, NUM_DD, RDWR, READ, WRITE

HDF_SD_ADDDATA procedure

positional arguments: 2

keywords: COUNT, START, STRIDE

HDF_SD_ATTRFIND() function

positional arguments: 2

keywords: none

HDF_SD_ATTRINFO procedure

positional arguments: 2

keywords: COUNT, DATA, HDF_TYPE, NAME, TYPE

HDF_SD_CREATE() function

positional arguments: 3

keywords: BYTE, DFNT_CHAR, DFNT_FLOAT32, DFNT_FLOAT64, DFNT_INT16, DFNT_INT32, DFNT_INT8, DFNT_UINT16, DFNT_UINT32, DFNT_UINT8, DOUBLE, FLOAT, HDF_TYPE, INT, LONG, SHORT, STRING

HDF_SD_DIMGET procedure

positional arguments: 1

keywords: COUNT, NAME, NATTR, SCALE

HDF_SD_DIMGETID() function

positional arguments: 2

keywords: none

HDF_SD_END procedure

positional arguments: 1

keywords: none

HDF_SD_ENDACCESS procedure

positional arguments: 1

keywords: none

HDF_SD_FILEINFO procedure

positional arguments: 3

keywords: none

HDF_SD_GETDATA procedure

positional arguments: 2

keywords: COUNT, START, STRIDE

HDF_SD_GETINFO procedure

positional arguments: 1

keywords: COORDSYS, DIMS, FORMAT, HDF_TYPE, LABEL, NAME, NATTS, NDIMS, TYPE, UNIT

HDF_SD_NAMETOINDEX() function

positional arguments: 2

keywords: none

HDF_SD_SELECT() function

positional arguments: 2

keywords: none

HDF_SD_START() function

positional arguments: 2

keywords: CREATE, RDWR, READ

HDF_VD_ATTACH() function

positional arguments: 2

keywords: READ, WRITE

HDF_VD_DETACH procedure

positional arguments: 1

keywords: none

HDF_VD_FIND() function

positional arguments: 2

keywords: none

HDF_VD_GET procedure

positional arguments: 1

keywords: CLASS, COUNT, NAME, REF, TAG

HDF_VD_READ() function

positional arguments: 2

keywords: FIELDS, FULL_INTERLACE, NO_INTERLACE, NRECORDS

HDF_VG_ATTACH() function

positional arguments: 2

keywords: READ, WRITE

HDF_VG_DETACH procedure

positional arguments: 1

keywords: none

HDF_VG_GETID() function

positional arguments: 2

keywords: none

HDF_VG_GETINFO procedure

positional arguments: 1

keywords: CLASS, NAME, NENTRIES, REF, TAG

HDF_VG_GETTRS procedure

positional arguments: 3

keywords: none

HEAP_GC procedure

positional arguments: none

keywords: OBJ, PTR, VERBOSE

HELP procedure

positional arguments: any number

keywords: BRIEF, CALLS, FUNCTIONS, INFO, LIB, MEMORY, OUTPUT, PROCEDURES, RECALL_COMMANDS, ROUTINES, STRUCTURES

HELPFORM() function

positional arguments: 2

keywords: FULL_STRUCT, SHORTFORM, SINGLE, SIZE, STRUCTURE_NAME, TAGFORM, WIDTH

HISTOGRAM() function

positional arguments: 1

keywords: BINSIZE, INPUT, LOCATIONS, MAX, MIN, NAN, NBINS, OMAX, OMIN, REVERSE_INDICES

HIST_2D() function

positional arguments: 2

keywords: BIN1, BIN2, MAX1, MAX2, MIN1, MIN2

implementation details: this routine is implemented as a wrapper to the HIST_ND() function

HIST_ND() function

positional arguments: 2

keywords: MAX, MIN, NBINS, REVERSE_INDICES

Performs an N-dimensional histogram, also known as the joint density function of N variables.

The first argument is an $N \times P$ array representing P data points in N dimensions. The second argument is optional, and it may be used to specify the size of the bin to use. Either an N point vector specifying a separate size for each dimension, or a scalar, which will be used for all dimensions. If BINSIZE is not passed, the NBINS keyword must be set (see below).

The function returns the N-Dimensional histogram, an array of size $N1 \times N2 \times N3 \times \dots \times ND$ where the N_i 's are the number of bins implied by the data, and/or the optional inputs (see below).

MIN keyword

The minimum value for the histogram. Either a P point vector specifying a separate minimum for each dimension, or a scalar, which will be used for all dimensions. If omitted, the natural minimum within the dataset will be used.

MAX keyword

The maximum value for the histogram. Either a P point vector specifying a separate maximum for each dimension, or a scalar, which will be used for all dimensions. If omitted, the natural maximum within the dataset will be used.

NBINS keyword

Rather than specifying the binsize, you can pass NBINS, the number of bins in each dimension, which can be a P point vector, or a scalar. If BINSIZE it also passed, NBINS will be ignored, otherwise BINSIZE will then be calculated as $\text{binsize} = (\text{max} - \text{min}) / \text{nbins}$.

REVERSE_INDICES keyword

Set to a named variable to receive the reverse indices, for mapping which points occurred in a given bin. Note that this is a 1-dimensional reverse index vector (see HISTOGRAM()). E.g., to find the indices of points which fell in a histogram bin [i,j,k], look up:

```
ind=[i+nx*(j+ny*k)]
ri[ri[ind]:ri[ind+1]-1]
```

See also ARRAY_INDICES() for converting in the other direction.

see also: [HISTOGRAM\(\)](#), [HIST_2D\(\)](#)

disclaimer: Entry based on J.D. Smith's documentation for his implementation of HIST_ND which was included in GDL unchanged. Copyright (C) 2001-2007, J.D. Smith. This software is provided as is without any warranty whatsoever. Permission to use, copy, modify, and distribute modified or unmodified copies is granted, provided this copyright and disclaimer are included unchanged.

IDENTITY() function

positional arguments: 1

keywords: [DOUBLE](#)

IDL_BASE64() function

positional arguments: 1

keywords: none

disclaimer: the name of this GDL routine includes the **IDL_** prefix for compatibility with IDL, it has no ...

IDL_VALIDNAME() function

positional arguments: 1

keywords: [CONVERT_ALL](#), [CONVERT_SPACES](#), [HELP](#), [TEST](#)

IGAMMA() function

positional arguments: 2

keywords: [DOUBLE](#)

IMAGE_STATISTICS procedure

positional arguments: 1

keywords: [COUNT](#), [DATA_SUM](#), [HELP](#), [LUT](#), [MASK](#), [MAXIMUM](#), [MEAN](#), [MINIMUM](#), [STDDEV](#), [SUM_OF_SQUARES](#), [TEST](#), [VARIANCE](#), [VECTOR](#), [VERBOSE](#), [WEIGHTED](#), [WEIGHT_SUM](#)

IMAGINARY() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

IMSL_BINOMIALCOEF() function

positional arguments: 2

keywords: [DOUBLE](#)

Returns the binomial coefficient defined as:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \text{ for } 0 \leq k \leq n \quad (15.6)$$

where n and k are the first and second arguments, respectively.

The routine can be used for example to construct the Pascal's triangle:

```
1 $ cat pascal.pro
2 pascal , 8
```

```
pro pascal , n
  tri = replicate(' ', 2 * n - 1, n)
  for i=0, n-1 do for j=0, i do tri[2*j + (n-i)-1, i] = $
    string(imsl_binomialcoef(i, j), f='(I3)')
  print, tri
end
```

% Compiled module: PASCAL.

```

                                     1
                                1   1
                           1   2   1
                      1   3   3   1
                1   4   6   4   1
            1   5   10  10  5   1
        1   6   15  20  15  6   1
    1   7   21  35  35  21  7   1
1   1   7   21  35  35  21  7   1
```

DOUBLE keyword

Forces double precision:

```
1 help, imsl_binomialcoef(1000, 20)
2 help, imsl_binomialcoef(1000, 20, /double)
```

<Expression>	FLOAT	=	inf
<Expression>	DOUBLE	=	3.3948281e+41

implementation details: this routine is a wrapper to the GSL's `gsl_sf_choose()` function [2]

disclaimer: the name of this GDL routine includes the **IMSL_** prefix for compatibility with IDL, it has no ...

IMSL_CONSTANT() function

positional arguments: 2

keywords: **DOUBLE**

```
1 print, 'Unified atomic mass, amu. [kg]:', '$'
2   imsl_constant('amu')
3 print, 'Pressure of 1 standard atmosphere [Pa]:', '$'
4   imsl_constant('atm')
5 print, ' -||- ', '$'
6   imsl_constant('StandardPressure')
7 print, 'Astronomical unit [m]:', '$'
8   imsl_constant('AU')
9 print, 'Avogadro's number [1/mole]:', '$'
10  imsl_constant('Avogadro')
11 print, 'Boltzmann constant [J/K]:', '$'
12  imsl_constant('Boltzman')
13 print, 'Speed of light in vacuum [m/s]:', '$'
14  imsl_constant('C')
15 print, ' -||- ', '$'
16  imsl_constant('Speedlight')
17 print, 'Base of the natural logarithm [1]:', '$'
18  imsl_constant('E')
19 print, 'Charge of the electron [C]:', '$'
```

```
20  imsl_constant('ElectronCharge')
21 print, 'Mass of the electron [kg]:', '$'
22  imsl_constant('ElectronMass')
23 print, 'The energy of 1 electron volt, eV [J]:', '$'
24  imsl_constant('ElectronVolt')
25 print, 'Euler-Mascheroni (gamma) constant [1]:', '$'
26  imsl_constant('Euler')
27 print, ' -||- ', '$'
28  imsl_constant('Gamma')
29 print, 'Molar charge of 1 Faraday [C/mole]:', '$'
30  imsl_constant('Faraday')
31 print, 'Electromagnetic fine structure constant [1]:', '$'
32  imsl_constant('FineStructure')
33 print, 'The molar gas constant [J/mole/K]:', '$'
34  imsl_constant('Gas')
35 print, 'The gravitational constant [N*m2/kg2]:', '$'
36  imsl_constant('Gravity')
37 print, 'Planck's constant divided by 2 pi [J*s]:', '$'
38  imsl_constant('Hbar')
39 print, 'The standard gas volume [m3 / mole]:', '$'
40  imsl_constant('PerfectGasVolume')
41 print, 'Pi [1]:', '$'
42  imsl_constant('Pi')
43 print, 'Planck's constant [J*s]:', '$'
44  imsl_constant('Planck')
45 print, 'Mass of the proton [kg]:', '$'
46  imsl_constant('ProtonMass')
47 print, 'Rydberg's constant [1/m]:', '$'
48  imsl_constant('Rydberg')
49 print, 'Standard gravitational acc. on Earth [m/s2]:', '$'
50  imsl_constant('StandardGravity')
51 print, 'Stefan-Boltzmann radiation const. [W/K4/m2]:', '$'
52  imsl_constant('StefanBoltzman')
53 print, 'Triple point temperature for water [K]:', '$'
54  imsl_constant('WaterTriple')
```

Unified atomic mass, amu. [kg]:	1.66054e-27
Pressure of 1 standard atmosphere [Pa]:	101325.
- -	101325.
Astronomical unit [m]:	1.49598e+11
Avogadro's number [1/mole]:	6.02214e+23

Boltzmann constant [J/K]:	1.38065e-23
Speed of light in vacuum [m/s]:	2.99792e+08
- -	2.99792e+08
Base of the natural logarithm [1]:	2.71828
Charge of the electron [C]:	1.60218e-19
Mass of the electron [kg]:	9.10938e-31
The energy of 1 electron volt, eV [J]:	1.60218e-19
Euler-Mascheroni (gamma) constant [1]:	0.57722
- -	0.57722
Molar charge of 1 Faraday [C/mole]:	96485.3
Electromagnetic fine structure constant [1]:	0.00730
The molar gas constant [J/mole/K]:	8.31447
The gravitational constant [N*m2/kg2]:	6.67300e-11
Planck's constant divided by 2 pi [J*s]:	1.05457e-34
The standard gas volume [m3 / mole]:	0.02271
Pi [1]:	3.14159
Planck's constant [J*s]:	6.62607e-34
Mass of the proton [kg]:	1.67262e-27
Rydberg's constant [1/m]:	1.09737e+07
Standard gravitational acc. on Earth [m/s2]:	9.80665
Stefan-Boltzmann radiation const. [W/K4/m2]:	5.67040e-08
Triple point temperature for water [K]:	273.160

implementation details: this routine uses the GSL's constants catalogue [2], the unit conversion is implemented using the UDUNITS-2 library

disclaimer: the name of this GDL routine includes the **IMSL_** prefix for compatibility with IDL, it has no ...

IMSL_ERF() function

positional arguments: 1

keywords: **DOUBLE**, **INVERSE**

IMSL_ZEROPOLY() function

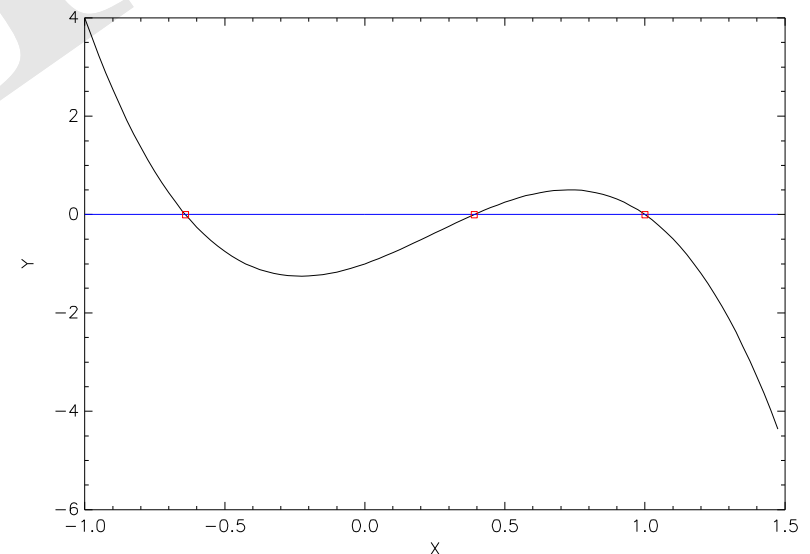
positional arguments: 1

keywords: **COMPANION**, **DOUBLE**, **JENKINS_TRAUB**

```

1 c = [-1,2,3,-4]
2 x = -1 + findgen(100) / 40
3 device, /color, /decomposed
4 plot, x, c[0] + c[1] * x + c[2] * x^2 + c[3] * x^3, $
5   xtitle='X', ytitle='Y', thick=3
6 oplot, x, replicate(0,n_elements(x)), color='ff0000' x
7 foreach z, imsl_zeropoly(c) do $
8   plots, z, 0., psym=6, thick=3, color='0000ff' x

```



implementation details: this routine is a wrapper to the GSL's `gsl_poly_complex_solve()` function [2]

disclaimer: the name of this GDL routine includes the **IMSL_** prefix for compatibility with IDL, it has no ...

IMSL_ZEROSYS() function

positional arguments: 2

keywords: **DOUBLE**, **ERR_REL**, **FNORM**, **ITMAX**, **JACOBIAN**, **XGUESS**

INDGEN() function

positional arguments: 8

keywords: **BYTE**, **COMPLEX**, **DCOMPLEX**, **DOUBLE**, **FLOAT**, **L64**, **LONG**, **STRING**,
TYPE, **UINT**, **UL64**, **ULONG**

INTARR() function

positional arguments: 8

keywords: **NOZERO**

INTERPOL() function

positional arguments: 3

keywords: **LSQUADRATIC**, **QUADRATIC**, **SPLINE**

INTERPOLATE() function

positional arguments: 4

keywords: **CUBIC**, **GRID**, **MISSING**

INVERT() function

positional arguments: 2

keywords: **DOUBLE**

ISHFT() function

positional arguments: 2

keywords: **_EXTRA**

JOURNAL procedure

positional arguments: 1

keywords: none

KEYWORD_SET() function

positional arguments: 1

keywords: none

KURTOSIS() function

positional arguments: 1

keywords: **DOUBLE**, **NAN**

L64INDGEN() function

positional arguments: 8

keywords: none

LAGUERRE() function

positional arguments: 3

keywords: **COEFFICIENTS**, **DOUBLE**

LAST_ITEM() function

positional arguments: 1

keywords: none

LA_TRIRED procedure

positional arguments: 3

keywords: **DOUBLE**, **UPPER**

LEGENDRE() function

positional arguments: 3

keywords: **DOUBLE****LINDGEN() function**

positional arguments: 8

keywords: none

LINKIMAGE procedure

positional arguments: 4

keywords: none

see also: **CALL_EXTERNAL()****LL_ARC_DISTANCE() function**

positional arguments: 3

keywords: **DEGREES**

Snyder [eqs. 5-5 and 5-6 in 5]

LMGR() function

positional arguments: none

keywords: **CLIENTSERVER, DEMO, EMBEDDED, EXPIRE_DATE, FORCE_DEMO, INSTALL_NUM, LMHOSTID, RUNTIME, SITE_NOTICE, STUDENT, TRIAL, VM****LNGAMMA() function**

positional arguments: 1

keywords: **DOUBLE****LOADCT procedure**

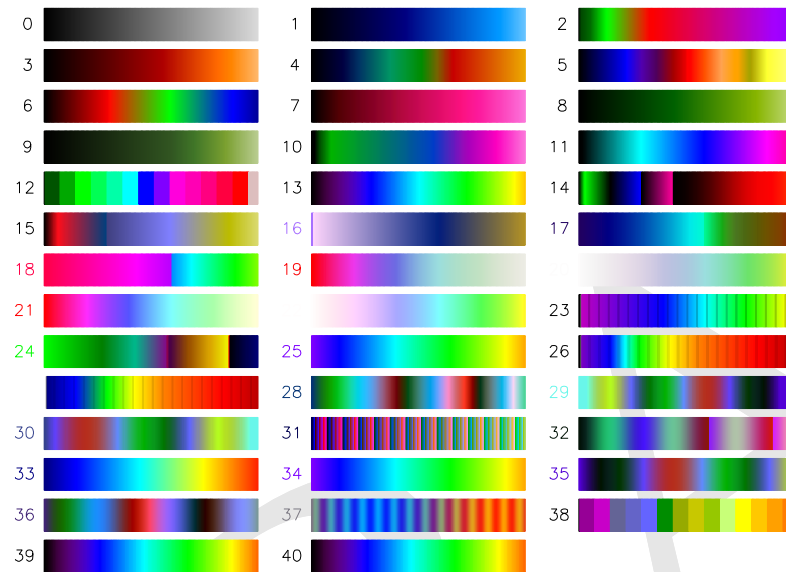
positional arguments: 1

keywords: **BOTTOM, FILE, GET_NAMES, NCOLORS, SILENT**

Loads a colour table that defines the RGB values corresponding to given colour indices (used when a plotting terminal is not set to the decomposed mode). The first argument may be used to choose from one of the 41 predefined colour tables, see example below for a graphical list of the colour predefined tables.

```
1 $ cat listct.pro
2 listct
```

```
pro listct
!X.STYLE=5
!Y.STYLE=5
!P.MULTI=[0,3,14]
!X.MARGIN=[10,0]
!Y.MARGIN=[1,0]
device, /color
for i=0, 40 do begin
    loadct, i, /silent
    contour, [[indgen(255)], [indgen(255)]], nlevels=256, /fill
    xyouts, -77, .5, strmid(i, 2)
endfor
end
% Compiled module: LISTCT.
% Compiled module: LOADCT.
```

**GET_NAMES keyword**

When set to a variable, a list of colour table names (string array) is assigned to that variable.

```
1 loadct, get_names=names
2 for i=0, n_elements(names)-1 do $
3   print, i, names[i], format='% "%d: %s"'
```

```
% Compiled module: LOADCT.
0: B-W LINEAR
1: BLUE/WHITE
2: GRN-RED-BLU-WHT
3: RED TEMPERATURE
4: BLUE/GREEN/RED/YELLOW
5: STD GAMMA-II
6: PRISM
7: RED-PURPLE
8: GREEN/WHITE LINEAR
9: GRN/WHT EXPONENTIAL
10: GREEN-PINK
11: BLUE-RED
```

```
12: 16 LEVEL
13: RAINBOW
14: STEPS
15: BOW SPECIAL
16: Haze
17: Blue — Pastel — Red
18: Pastels
19: Hue Sat Lightness 1
20: Hue Sat Lightness 2
21: Hue Sat Value 1
22: Hue Sat Value 2
23: Purple-Red + Stripes
24: Beach
25: Mac Style
26: Eos A
27: Eos B
28: Hardcandy
29: Nature
30: Ocean
31: Peppermint
32: Plasma
33: Blue-Red
34: Rainbow
35: Blue Waves
36: Volcano
37: Waves
38: Rainbow18
39: Rainbow + white
40: Rainbow + black
```

LOADCT_INTERNALGDL procedure

positional arguments: 1

keywords: **GET_NAMES**

LOCALE_GET() function

positional arguments: none

LOGICAL_AND() function

keywords: none

LOGICAL_AND() function

positional arguments: 2

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

LOGICAL_OR() function

positional arguments: 2

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

LOGICAL_TRUE() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

LON64ARR() function

positional arguments: 8

keywords: NOZERO

LONARR() function

positional arguments: 8

keywords: NOZERO

LONG() function

positional arguments: 10

keywords: none

LONG64() function

positional arguments: 10

keywords: none

LUDC procedure

positional arguments: 2

keywords: COLUMN, DOUBLE, INTERCHANGES

LUSOL() function

positional arguments: 3

keywords: COLUMN, DOUBLE

MACHAR() function

positional arguments: none

keywords: DOUBLE

MAGICK_ADDNOISE procedure

positional arguments: 1

keywords: GAUSSIANNNOISE, IMPULSENNOISE, LAPLACIANNNOISE, MULTIPLICATIVE-GAUSSIANNNOISE, NOISE, POISSONNOISE, UNIFORMNOISE

MAGICK_CLOSE procedure

positional arguments: 1

keywords: none

MAGICK_COLORMAPSIZE() function

positional arguments: 2

keywords: none

MAGICK_COLUMNS() function

positional arguments: 1

keywords: none

MAGICK_CREATE() function

positional arguments: 3

keywords: none

MAGICK_DISPLAY procedure

positional arguments: 1

keywords: none

MAGICK_EXISTS() function

positional arguments: none

keywords: none

MAGICK_FLIP procedure

positional arguments: 1

keywords: none

MAGICK_INDEXEDCOLOR() function

positional arguments: 1

keywords: none

MAGICK_INTERLACE procedure

positional arguments: 1

keywords: **LINEINTERLACE**, **NOINTERLACE**, **PLANEINTERLACE**

MAGICK_MAGICK() function

positional arguments: 2

keywords: none

MAGICK_MATTE procedure

positional arguments: 1

keywords: none

MAGICK_OPEN() function

positional arguments: 1

keywords: none

MAGICK_PING() function

positional arguments: 2

keywords: **CHANNELS**, **DIMENSIONS**, **GAUSSIANNNOISE**, **HAS_PALETTE**, **IMAGE_INDEX**, **IMPULSENNOISE**, **INFO**, **LAPLACIANNNOISE**, **MULTIPLICATIVEGAUSSIANNNOISE**, **NOISE**, **NUM_IMAGES**, **PIXEL_TYPE**, **POISSONNOISE**, **TYPE**, **UNIFORMNOISE**

MAGICK_QUALITY procedure

positional arguments: 2

keywords: none

MAGICK_QUANTIZE procedure

positional arguments: 2

keywords: DITHER, GRAYSCALE, TRUECOLOR, YUV

MAGICK_READ() function

positional arguments: 1

keywords: MAP, RGB, SUB_RECT

MAGICK_READCOLORMAPRGB procedure

positional arguments: 4

keywords: none

MAGICK_READINDEXES() function

positional arguments: 1

keywords: none

MAGICK_ROWS() function

positional arguments: 1

keywords: none

MAGICK_WRITE procedure

positional arguments: 2

keywords: RGB

MAGICK_WRITECOLORTABLE procedure

positional arguments: 4

keywords: none

MAGICK_WRITEFILE procedure

positional arguments: 3

keywords: none

MAGICK_WRITEINDEXES procedure

positional arguments: 2

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

MAKE_ARRAY() function

positional arguments: 8

keywords: BYTE, COMPLEX, DCOMPLEX, DIMENSION, DOUBLE, FLOAT, INDEX, INTEGER, L64, LONG, NOZERO, OBJ, PTR, SIZE, STRING, TYPE, UINT, UL64, ULONG, VALUE

MAP_CLIP_SET procedure

positional arguments: none

keywords: CLIP_PLANE, CLIP_UV, RESET, SPLIT, TRANSFORM

MAP_CONTINENTS procedure

positional arguments: none

keywords: COLOR, COUNTRIES, FILL_CONTINENTS, HIRES, RIVERS

Wessel and Smith [7]

MATRIX_MULTIPLY() function

positional arguments: 2

keywords: ATRANSPOSE, BTRANSPOSE

MAX() function

positional arguments: 2

keywords: **DIMENSION**, **MIN**, **NAN**, **SUBSCRIPT_MIN**

MEAN() function

positional arguments: 1

keywords: **DOUBLE**, **NAN**

MEANABSDEV() function

positional arguments: 1

keywords: **DOUBLE**, **NAN**

MEDIAN() function

positional arguments: 2

keywords: **DIMENSION**, **DOUBLE**, **EVEN**

MEMORY() function

positional arguments: 1

keywords: **CURRENT**, **HIGHWATER**, **L64**, **NUM_ALLOC**, **NUM_FREE**, **STRUCTURE**

MESSAGE procedure

positional arguments: 1

keywords: **CONTINUE**, **INFORMATIONAL**, **IOERROR**, **NONAME**, **NOPREFIX**, **NO-PRINT**, **RESET**, **TRACEBACK**

MIN() function

positional arguments: 2

keywords: **DIMENSION**, **MAX**, **NAN**, **SUBSCRIPT_MAX**

MOMENT() function

positional arguments: 1

keywords: **DOUBLE**, **MAXMOMENT**, **MDEV**, **NAN**, **SDEV**

NCDF_ATTCOPY() function

positional arguments: 5

keywords: **IN_GLOBAL**, **OUT_GLOBAL**

NCDF_ATTDEL procedure

positional arguments: 3

keywords: **GLOBAL**

NCDF_ATTGET procedure

positional arguments: 4

keywords: **GLOBAL**

NCDF_ATTINQ() function

positional arguments: 3

keywords: **GLOBAL**

NCDF_ATTNAME() function

positional arguments: 3

keywords: **GLOBAL**

NCDF_ATTPUT procedure

positional arguments: 4

keywords: **BYTE**, **CHAR**, **DOUBLE**, **FLOAT**, **GLOBAL**, **LENGTH**, **LONG**, **SHORT**

NCDF_ATTRENAME procedure

positional arguments: 4

keywords: GLOBAL

NCDF_CLOSE procedure

positional arguments: 1

keywords: none

NCDF_CONTROL procedure

positional arguments: 1

keywords: ABORT, ENDEF, FILL, NOFILL, NOVERBOSE, OLDFILL, REDEF, SYNC, VERBOSE

NCDF_CREATE() function

positional arguments: 1

keywords: CLOBBER, NOCLOBBER

NCDF_DIMDEF() function

positional arguments: 3

keywords: UNLIMITED

NCDF_DIMID() function

positional arguments: 2

keywords: none

NCDF_DIMINQ procedure

positional arguments: 4

keywords: none

NCDF_DIMRENAME procedure

positional arguments: 3

keywords: none

NCDF_EXISTS() function

positional arguments: none

keywords: none

```
1 print, 'GDL compiled with netCDF support: ' $  
2   + (ncdf_exists() ? 'yes' : 'no')
```

```
GDL compiled with netCDF support: yes
```

NCDF_INQUIRE() function

positional arguments: 1

keywords: none

NCDF_OPEN() function

positional arguments: 1

keywords: NOWRITE, WRITE

NCDF_VARDEF() function

positional arguments: 3

keywords: BYTE, CHAR, DOUBLE, FLOAT, LONG, SHORT

NCDF_VARGET procedure

positional arguments: 3

keywords: COUNT, OFFSET, STRIDE

NCDF_VARGET1 procedure

positional arguments: 3

keywords: **OFFSET**

NCDF_VARID() function

positional arguments: 2

keywords: none

NCDF_VARINQ() function

positional arguments: 2

keywords: none

NCDF_VARPUT procedure

positional arguments: 3

keywords: **COUNT**, **OFFSET**, **STRIDE**

NCDF_CONTROL with SYNC to force...

NCDF_VARRENAME procedure

positional arguments: 3

keywords: none

NEWTON() function

positional arguments: 2

keywords: **DOUBLE**, **HYBRID**, **ITMAX**, **TOLF**, **TOLX**

Galassi et al. [2]

NORM() function

positional arguments: 1

keywords: **DOUBLE**

N_ELEMENTS() function

positional arguments: 1

keywords: none

N_PARAMS() function

positional arguments: 1

keywords: none

N_TAGS() function

positional arguments: 1

keywords: **DATA_LENGTH**, **LENGTH**

OBJARR() function

positional arguments: 8

keywords: **NOZERO**

OBJ_CLASS() function

positional arguments: 1

keywords: **COUNT**, **SUPERCLASS**

Returns the name of the class of an object passed as the first argument.

SUPERCLASS keyword

Returns instead an array of all direct superclasses of the object passed as the first argument. In this case the first argument may be a string defining the object name.

COUNT keyword

Allows to pass a reference to a variable into which the number of direct superclasses will be stored.

```

1 $ tail *__define.pro
2 bottle = obj_new('beer')
3 print, 'bottle is a[n] ', obj_class(bottle)
4 spr = obj_class('beer', /superclass, count=cnt)
5 print, 'beer has ', strtrim(cnt,2) , ' direct superclass[es]: ', strjoin(spr, ',')
```

```

=> alcoholic_drink__define.pro <==
pro alcoholic_drink__define
  struct = {alcoholic_drink, proof : 0, inherits drink}
end
```

```

=> beer__define.pro <==
pro beer__define
  struct = {beer, inherits alcoholic_drink}
end
```

```

=> drink__define.pro <==
pro drink__define
  struct = {drink, color : 0}
end
% Compiled module: BEER__DEFINE.
% Compiled module: ALCOHOLIC_DRINK__DEFINE.
% Compiled module: DRINK__DEFINE.
bottle is a[n] BEER
beer has 1 direct superclass[es]: ALCOHOLIC_DRINK
```

A list of all known classes is returned if called without any argument:

```

1 classes = obj_class()
2 help, classes
3 print, classes
```

```

CLASSES      STRING      = Array[24]
!PLT !GNUDataLanguage !AXIS !VERSION !MOUSE !ERROR_STATE !VALUES !MAP !CPU !WARN !USERSYM !DL_SIZE !FSTAT64 !FSTAT !FILE_INFO !IDL_MEMORY
IDL_MEMORY64 MACHAR DMACHAR WIDGET_BUTTON WIDGET_DROPLIST WIDGET_TEXT WIDGET_VERSION !DEVICE
```

OBJ_DESTROY procedure

positional arguments: any number

keywords: **_REF_EXTRA**

OBJ_ISA() function

positional arguments: 2

keywords: none

OBJ_NEW() function

positional arguments: any number

keywords: **_REF_EXTRA**

Beware that values of object fields may only be initialised in the constructor, and not while defining the object structure, i.e.:

```

1 $ cat test__define.pro
2 a = obj_new('test ')
3 a->printXY
```

```

pro test::printXY
  print, self.x, self.y
end
function test::init
  self.x = 10
  return, 1
end
pro test__define
  struct = {test, x : 5, y : 5}
end
% Compiled module: TEST__DEFINE
!MAP !CPU !WARN !USERSYM !DL_SIZE !FSTAT64 !FSTAT !FILE_INFO !IDL_MEMORY
IDL_MEMORY64 MACHAR DMACHAR WIDGET_BUTTON WIDGET_DROPLIST WIDGET_TEXT WIDGET_VERSION !DEVICE
```

OBJ_VALID() function

positional arguments: 1

keywords: **CAST**, **COUNT**

ON_ERROR procedure

positional arguments: 1

keywords: none

OPENR procedure

positional arguments: 3

keywords: **APPEND**, **BINARY**, **BLOCK**, **BUFSIZE**, **COMPRESS**, **DELETE**, **ERROR**, **F77_UNFORMATTED**, **GET_LUN**, **MORE**, **NOAUTOMODE**, **STDIO**, **STREAM**, **SWAP_ENDIAN**, **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**, **VAX_FLOAT**, **WIDTH**, **XDR**

COMPRESS keyword

```

1 $ echo "GDL rocks!" > file.txt
2 $ gzip -f file.txt
3 openr, u, 'file.txt.gz', /get_lun, /compress
4 s = '
5 readu, u, s
6 free_lun, u
7 print, s
8 $ rm file.txt.gz

```

```
GDL rocks!
```

OPENU procedure

positional arguments: 3

keywords: **APPEND**, **BINARY**, **BLOCK**, **BUFSIZE**, **COMPRESS**, **DELETE**, **ERROR**, **F77_UNFORMATTED**, **GET_LUN**, **MORE**, **NOAUTOMODE**, **STDIO**, **STREAM**, **SWAP_ENDIAN**, **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**, **VAX_FLOAT**, **WIDTH**, **XDR**

OPENW procedure

positional arguments: 3

keywords: **APPEND**, **BINARY**, **BLOCK**, **BUFSIZE**, **COMPRESS**, **DELETE**, **ERROR**, **F77_UNFORMATTED**, **GET_LUN**, **MORE**, **NOAUTOMODE**, **STDIO**, **STREAM**, **SWAP_ENDIAN**, **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**, **VAX_FLOAT**, **WIDTH**, **XDR**

OPLOT procedure

positional arguments: 2

keywords: **CLIP**, **COLOR**, **LINESTYLE**, **MAX_VALUE**, **MIN_VALUE**, **NOCLIP**, **NSUM**, **POLAR**, **PSYM**, **SYMSIZE**, **T3D**, **THICK**

PARSE_URL() function

positional arguments: 1

keywords: none

Returns a structure describing components of the URL passed as an argument, e.g.:

```
1 help, parse_url('http://root:qwerty@kgb.ru:666/?hack'), /stru
```

```

** Structure <Anonymous>, 7 tags, data length=56:
SCHEME          STRING      'http'
USERNAME        STRING      'root'
PASSWORD        STRING      'qwerty'
HOST            STRING      'kgb.ru'
PORT            STRING      '666'
PATH            STRING      '/'
QUERY           STRING      'hack'

```

PATH_SEP() function

positional arguments: none

keywords: PARENT_DIRECTORY, SEARCH_PATH, TEST

PLOT procedure

positional arguments: 2

keywords: BACKGROUND, CHARSIZE, CHARTHICK, CLIP, COLOR, DATA, DEVICE, LINESSTYLE, MAX_VALUE, MIN_VALUE, NOCLIP, NODATA, NOERASE, NORMAL, POSITION, PSYM, SUBTITLE, SYMSIZE, THICK, TICKLEN, TITLE, XCHARSIZE, XLOG, XMARGIN, XMINOR, X RANGE, XSTYLE, XTHICK, XTICKFORMAT, XTICKLEN, XTICKS, XTITLE, XTYPE, YCHARSIZE, YLOG, YMARGIN, YMINOR, YNOZERO, YRANGE, YSTYLE, YTHICK, YTICKFORMAT, YTICKLEN, YTICKS, YTITLE, YTYPE, ZCHARSIZE, ZGRIDSTYLE, ZMARGIN, ZMINOR, ZRANGE, ZSTYLE, ZTHICK, ZTICKFORMAT, ZTICKLEN, ZTICKS, ZTITLE, ZVALUE

PLOTERR procedure

positional arguments: 4

keywords: BAR_COLOR, HAT, HELP, LENGTH_OF_HAT, PSYM, TEST, TYPE, XLOG, X RANGE, YLOG, YRANGE, _EXTRA

PLOTS procedure

positional arguments: 3

keywords: CLIP, COLOR, CONTINUE, DATA, DEVICE, LINESSTYLE, NOCLIP, NORMAL, PSYM, SYMSIZE, T3D, THICK

PM procedure

positional arguments: any number

keywords: FORMAT, TITLE

```
1 arr = indgen(4,4)
2 fmt = '(4I3)'
3 print, 'PM'
4 pm, arr, format=fmt
5 print, 'PRINT:'
6 print, arr, format=fmt
```

```
PM
0  4  8 12
1  5  9 13
2  6 10 14
3  7 11 15
PRINT:
0  1  2  3
4  5  6  7
8  9 10 11
12 13 14 15
```

see also: ORDER keyword in TV, TVRD(), ... (TODO: section on # and ## ops.)

POINT_LUN procedure

positional arguments: 2

keywords: none

POLY() function

positional arguments: 2

keywords: none

POLYFILL procedure

positional arguments: 3

keywords: CLIP, COLOR, DATA, DEVICE, LINESSTYLE, LINE_FILL, NOCLIP, NORMAL, ORIENTATION, SPACING, THICK

POLY_2D() function

positional arguments: 6

keywords: CUBIC, MISSING

POLY_AREA() function

positional arguments: 2

keywords: DOUBLE, SIGNED

POPD procedure

positional arguments: none

keywords: none

PREWITT() function

positional arguments: 1

keywords: HELP

PRIMES() function

positional arguments: 1

keywords: none

PRINT procedure

positional arguments: any number

keywords: AM_PM, DAYS_OF_WEEK, FORMAT, MONTH, STUDIO_NON_FINITE

PRINTD procedure

positional arguments: none

keywords: none

PRINTF procedure

positional arguments: any number

keywords: AM_PM, DAYS_OF_WEEK, FORMAT, MONTH, STUDIO_NON_FINITE

PRODUCT() function

positional arguments: 2

keywords: CUMULATIVE, INTEGER, NAN, PRESERVE_TYPE

multi-threading: this routine uses GDL thread pool if working on large array, see the...**PTRARR() function**

positional arguments: 8

keywords: ALLOCATE_HEAP, NOZERO

multi-threading: this routine uses GDL thread pool if working on large array, see the...**PTR_FREE procedure**

positional arguments: any number

keywords: none

PTR_FREE can also be used to deallocate a variable:

```

1 a = 1
2 ptr_free , ptr_new(a , /no_copy)
3 help , a

```

A	UNDEFINED = <Undefined>
---	-------------------------

see also: PTR_NEW(), PTR_VALID()

PTR_NEW() function

positional arguments: 1

keywords: **ALLOCATE_HEAP**, **NO_COPY****PTR_VALID() function**

positional arguments: 1

keywords: **CAST**, **COUNT****PUSHD procedure**

positional arguments: 1

keywords: none

PYTHON procedure

positional arguments: any number

keywords: **ARGV****PYTHON() function**

positional arguments: any number

keywords: **ARGV**, **DEFAULTRETURNVALUE**

Executes a python function whose name is specified using the second argument, the first argument defines the package (e.g. numpy). All other argument are passed as positional arguments to the function.

```
1 print, python('numpy', 'arange', 4.)
```

```
0.0000000 1.0000000 2.0000000 3.0000000
```

PY_PLOT procedure

positional arguments: 2

keywords: **GRID**, **TITLE**, **XLABEL**, **YLABEL****PY_PRINT procedure**

positional arguments: 1

keywords: none

QUERY_BMP() function

positional arguments: 2

keywords: none

```
1 $ wget --quiet http://wikipedia.org/favicon.ico
2 $ convert favicon.ico favicon.bmp
3 ok = query_bmp('favicon.bmp', info)
4 if ok then help, info, /structure else print, 'query failed!'
5 $ rm favicon.*
```

```
% Compiled module: QUERY_BMP.
** Structure <Anonymous>, 7 tags, data length=56:
CHANNELS          LONG              4
DIMENSIONS        LONG      Array[2]
HAS_PALETTE       INT                0
IMAGE_INDEX       LONG              0
NUM_IMAGES        LONG              1
PIXEL_TYPE        INT                1
TYPE              STRING      'BMP'
```

QUERY_DICOM() function

positional arguments: 2

keywords: none

QUERY_GIF() function

positional arguments: 2

keywords: none

QUERY_IMAGE() function

positional arguments: 2

keywords: `_REF_EXTRA`

QUERY_JPEG() function

positional arguments: 2

keywords: none

QUERY_PICT() function

positional arguments: 2

keywords: none

QUERY_PNG() function

positional arguments: 2

keywords: none

QUERY_PPM() function

positional arguments: 2

keywords: none

QUERY_TIFF() function

positional arguments: 2

keywords: `IMAGE_INDEX`

RADON() function

positional arguments: 1

keywords: `BACKPROJECT`, `DOUBLE`, `DRHO`, `DX`, `DY`, `GRAY`, `LINEAR`, `NRHO`, `NTHETA`, `NX`, `NY`, `RHO`, `RMIN`, `THETA`, `XMIN`, `YMIN`

RANDOMN() function

positional arguments: 8

keywords: `BINOMIAL`, `DOUBLE`, `GAMMA`, `LONG`, `NORMAL`, `POISSON`, `UNIFORM`

RANDOMU() function

positional arguments: 8

keywords: `BINOMIAL`, `DOUBLE`, `GAMMA`, `LONG`, `NORMAL`, `POISSON`, `UNIFORM`

READ procedure

positional arguments: any number

keywords: `AM_PM`, `DAYS_OF_WEEK`, `FORMAT`, `MONTH`, `PROMPT`

READF procedure

positional arguments: any number

keywords: `AM_PM`, `DAYS_OF_WEEK`, `FORMAT`, `MONTH`, `PROMPT`

READS procedure

positional arguments: any number

keywords: `AM_PM`, `DAYS_OF_WEEK`, `FORMAT`, `MONTH`

READU procedure

positional arguments: any number

keywords: `TRANSFER_COUNT`

READ_ASCII() function

positional arguments: 1

keywords: COMMENT_SYMBOL, COUNT, DATA_START, DELIMITER, HEADER, HELP, MISSING_VALUE, NUM_RECORDS, RECORD_START, TEMPLATE, TEST, VERBOSE

READ_BINARY() function

positional arguments: 1

keywords: DATA_DIMS, DATA_START, DATA_TYPE, ENDIAN, TEMPLATE

READ_BMP() function

positional arguments: 4

keywords: RGB

READ_DICOM() function

positional arguments: 4

keywords: IMAGE_INDEX

READ_GIF procedure

positional arguments: 5

keywords: DEBUG, HELP, TEST

READ_JPEG procedure

positional arguments: 3

keywords: BUFFER, COLORS, DEBUG, DITHER, GRAYSCALE, HELP, ORDER, TEST, TRUE, TWO_PASS_QUANTIZE, UNIT

READ_PICT procedure

positional arguments: 5

keywords: none

READ_PNG() function

positional arguments: 4

keywords: HELP, ORDER, TEST, TRANSPARENT, VERBOSE

READ_TIFF() function

positional arguments: 4

keywords: CHANNELS, GEOTIFF, IMAGE_INDEX, INTERLEAVE, ORIENTATION, PLANARCONFIG, SUB_RECT, VERBOSE

READ_XWD() function

positional arguments: 4

keywords: none

REAL_PART() function

positional arguments: 1

keywords: none

REBIN() function

positional arguments: 9

keywords: SAMPLE

RECALL_COMMANDS() function

positional arguments: none

keywords: none

REFORM() function

positional arguments: 8
keywords: **OVERWRITE**

REPLICATE() function

positional arguments: 9
keywords: none

REPLICATE_INPLACE procedure

positional arguments: 6
keywords: none

RESOLVE_ROUTINE procedure

positional arguments: 1
keywords: none

RESTORE procedure

positional arguments: 1
keywords: **DESCRIPTION**, **FILENAME**, **RELAXED_STRUCTURE_ASSIGNMENT**, **RESTORED_OBJECTS**, **VERBOSE**

RETALL procedure

positional arguments: none
keywords: **RETALL**

REVERSE() function

positional arguments: 2
keywords: **OVERWRITE**

RK4() function

positional arguments: 5
keywords: **DOUBLE**, **ITER**

RK4JMG() function

positional arguments: 5
keywords: **DOUBLE**

ROBERTS() function

positional arguments: 1
keywords: **HELP**

ROTATE() function

positional arguments: 2
keywords: none

ROUND() function

positional arguments: 1
keywords: **L64**

multi-threading: this routine uses GDL thread pool if working on large array, see the...

ROUTINE_INFO() function

positional arguments: 1
keywords: **DISABLED**, **ENABLED**, **FUNCTIONS**, **PARAMETERS**, **SYSTEM**

ROUTINE_NAMES() function

positional arguments: any number

keywords: **ARG_NAME**, **FETCH**, **LEVEL**, **STORE**, **S_FUNCTIONS**, **S_PROCEDURES**, **VARIABLES**

Examines variables and parameters of procedures and the call stack Using ROUTINE_NAMES a subroutine can interrogate, and in some cases change, the values and names of variables and parameters in its calling routine, or at the \$MAIN\$ level.

ROUTINE_NAMES uses a notion of the current "call level," which is the numerical stack depth of the currently executing routine. At each procedure or function call, the call level becomes one **deeper**, and upon each RETURN, the call level becomes one **shallower**. The call stack always begins at the \$MAIN\$ level. The current call stack can always be printed by executing **HELP**.

When specifying the call level to ROUTINE_NAMES, one can use one of two numbering systems, depending on whichever is most convenient. In the **absolute** numbering system, the \$MAIN\$ level starts at number 1, and becomes deeper with increasing numbers. In the **relative** numbering system, the current (deepest) call level is number 0, and becomes shallower with more negative numbers. Hence, if the deepest level is N, then the correspondence is thus:

VALUE	MEANING
1 or -N+1	\$MAIN\$ level
2 or -N+2	NEXT deeper level
...	...
N or 0	DEEPEST (currently executing) level

When called without any keyword ROUTINE_NAMES returns a string array containing a list of currently compiled functions and procedures, e.g.:

```
1 $ cat library.pro
2 .compile library.pro
3 print, routine_names()
```

```
pro a_procedure
  print, 'Hello world!'
end
function a_function
  return, 'Hello world!'
end
% Compiled module: A_PROCEDURE.
% Compiled module: A_FUNCTION.
```

```
$MAIN$ A_FUNCTION A_PROCEDURE
```

ROUTINE_NAMES can be invoked in several other ways, which are detailed below together with keyword descriptions.

S_PROCEDURES keyword

The lists of system procedures is returned, as a string array. The list does not cover procedures written in GDL itself which are also part of GDL's routine library (e.g. **WRITE_PNG**).

```
1 print, (routine_names(/s_pro))[0:5]
```

```
AXIS BYTEORDER CALDAT CALL_METHOD CALL_PROCEDURE CATCH
```

S_FUNCTIONS keyword

The lists of system functions is returned, as a string array. The list does not cover functions written in GDL itself which are also part of GDL's routine library (e.g. **READ_PNG()**).

```
1 help, routine_names(/s_functions)
```

```
<Expression>    STRING    = Array[250]
```

LEVEL keyword

The call level of the calling routine is returned, e.g.:

```
1 $ cat func.pro
2 print, routine_names(/level), func()
```

```
function func
  return, routine_names(/level)
end
% Compiled module: FUNC.
          1          2
```

ARG_NAME keyword

The names of variables passed as positional arguments at call level specified with the ARG_NAME keyword are returned, as a string array. Note that the arguments passed are the actual parameters, not strings containing their names. All of the arguments must be parameters that have been passed to the calling procedure. Variables that are unnamed at the specified call level will return the empty string.

```
1 $ cat procedure.pro
2 a1 = 1
3 a2 = '2'
4 a3 = [3b]
5 procedure, a1, a2, a3
```

```
pro procedure, arg0, arg1, arg2
  print, routine_names(arg1, arg2, arg_name=0)
  print, routine_names(arg1, arg2, arg_name=-1)
end
% Compiled module: PROCEDURE.
ARG1 ARG2
A2 A3
```

VARIABLES keyword

The names of variables at call level specified with the VARIABLES keyword are returned, as a string array, e.g.:

```
1 $ cat procedure.pro
2 str = 'Hello world!'
3 arr = ['Hello ', 'world ', '!']
4 int = 0
5 procedure
```

```
pro procedure
  print, routine_names(variables=-1)
end
% Compiled module: PROCEDURE.
STR ARR INT
```

FETCH keyword

The value of a variable which name is passed in the first argument (string) at call level specified with the FETCH keyword is returned. If the value is undefined, then the assignment will cause an error. Therefore, the only safe way to retrieve a value is by using a variant of the following:

```
1 if n_elements(routine_names('a', fetch=0)) gt 0 $
2   then value = routine_names('a', fetch=0) $
3   else message, 'a is not defined!'
```

```
% $MAIN$: a is not defined!
% Execution halted at: $MAIN$
```

STORE keyword

The value specified with the second argument is stored into the variable which name is passed in the first argument (string) at the call level specified with the STORE keyword. Note that there is no way to cause the named variable to become undefined. The value returned can be ignored.

```
1 a = 1
2 dummy = routine_names('a', 2, store=0)
3 print, a
```

```
2
```

see also: [ROUTINE_INFO\(\)](#), [ARG_PRESENT\(\)](#), [SCOPE_VARFETCH\(\)](#)

disclaimer: Entry based on Craig Markwardt's documentation for ROUTINE_NAMES:
Copyright (C) 2000, Craig Markwardt. This software is provided as is without any warranty whatsoever. Permission to use, copy, modify, and distribute modified or unmodified copies is granted, provided this copyright and disclaimer are included unchanged.

RSTRPOS() function

positional arguments: 3

keywords: none

SAVE procedure

positional arguments: 30

keywords: ALL, APPEND, COMPATIBLE, DATA, ERRMSG, FILENAME, MTIMES, NAMES, NOCATCH, PASS_METHOD, QUIET, STATUS, TEST, USEUNIT, VARSTATUS, VERBOSE, XDR

SCOPE_VARFETCH() function

positional arguments: 1

keywords: LEVEL

SEM_CREATE() function

positional arguments: 1

keywords: DESTROY_SEMAPHORE

SEM_DELETE procedure

positional arguments: 1

keywords: none

SEM_LOCK() function

positional arguments: 1

keywords: none

SEM_RELEASE procedure

positional arguments: 1

keywords: none

SETENV procedure

positional arguments: 1

keywords: none

SET_PLOT procedure

positional arguments: 1

keywords: COPY, INTERPOLATE

SHIFT() function

positional arguments: 9

keywords: none

SHOWFONT procedure

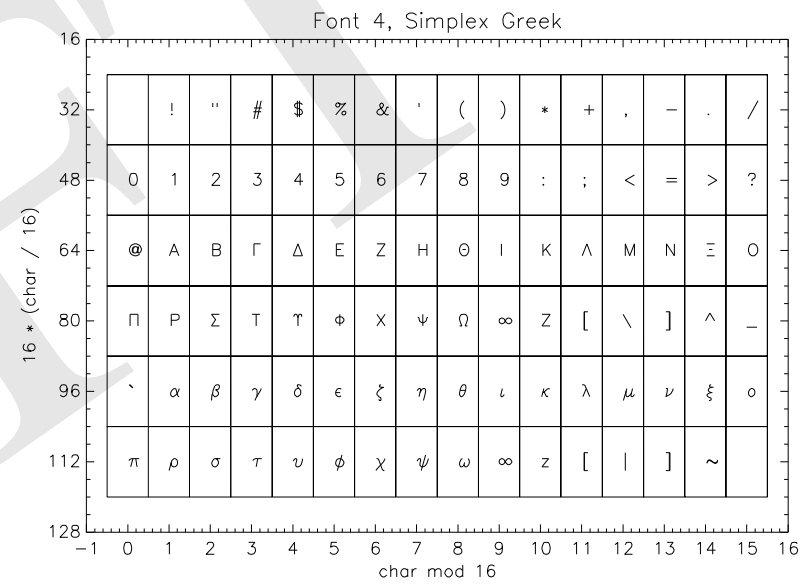
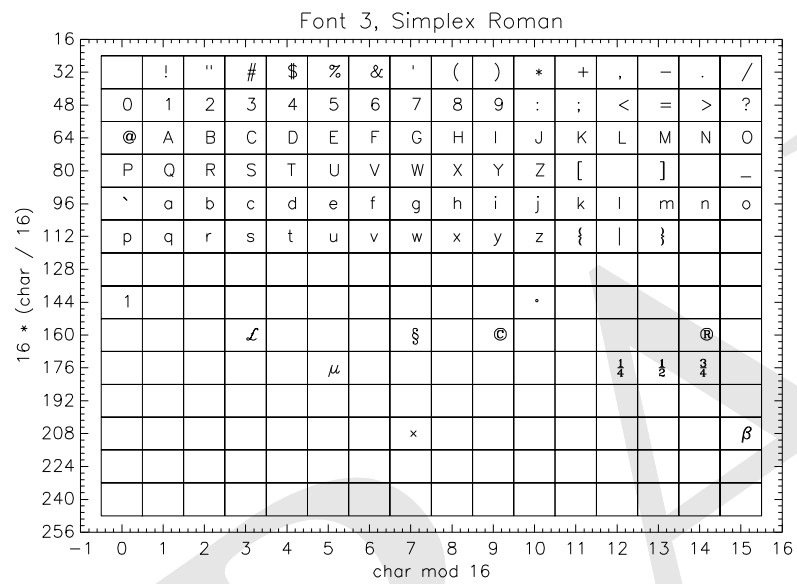
positional arguments: 2

keywords: BASE, BEG, ENCAPSULATED, FIN, TT_FONT

Displays a table of fonts for a give font number (first argument) in the current graphics terminal, e.g.:

```
1 showfont , 3, 'Simplex Roman'
```

```
% Compiled module: SHOWFONT.
```

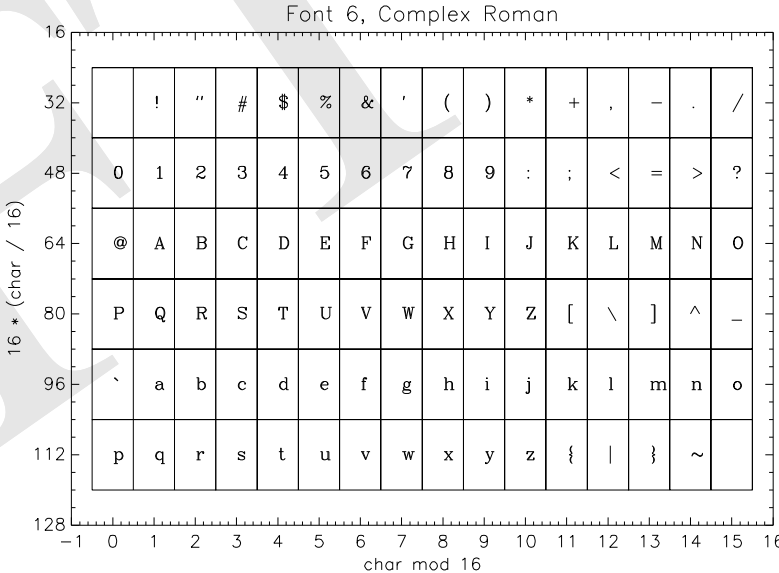
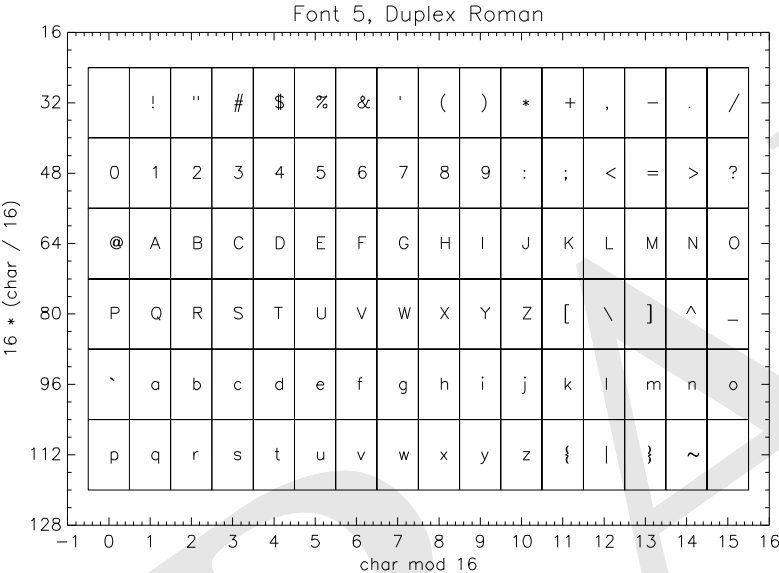


```
1 showfont , 4, 'Simplex Greek'
```

```
1 | showfont, 5, 'Duplex Roman'
```

```
% Compiled module: SHOWFONT.
```

```
% Compiled module: SHOWFONT.
```

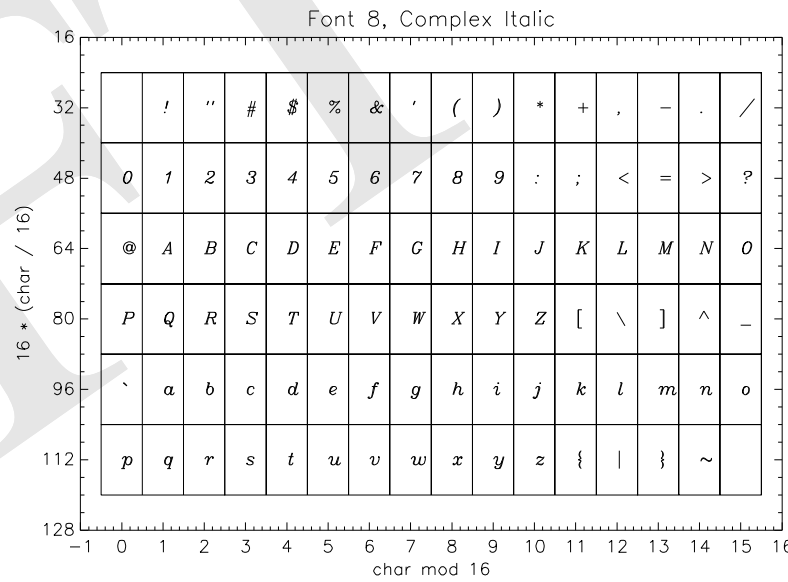
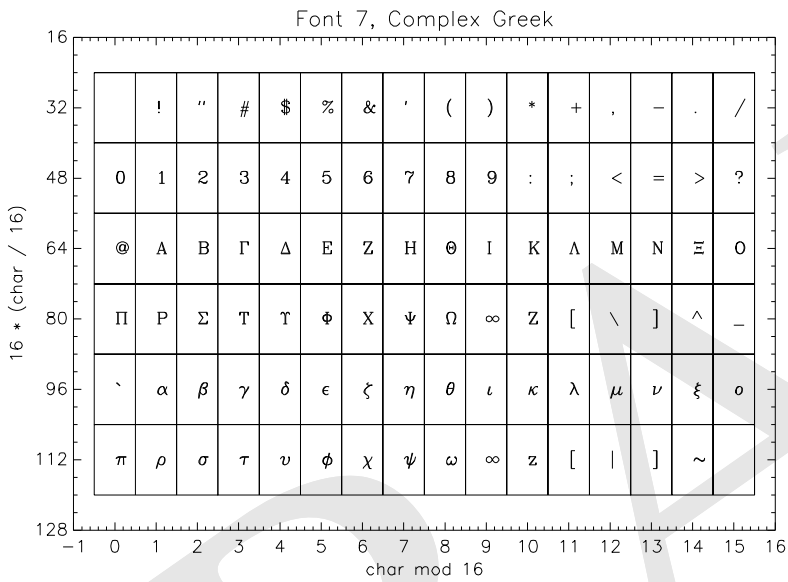


```
1 showfont , 6, 'Complex Roman'
```

```
1 showfont , 7, 'Complex Greek'
```

% Compiled module: SHOWFONT.

% Compiled module: SHOWFONT.

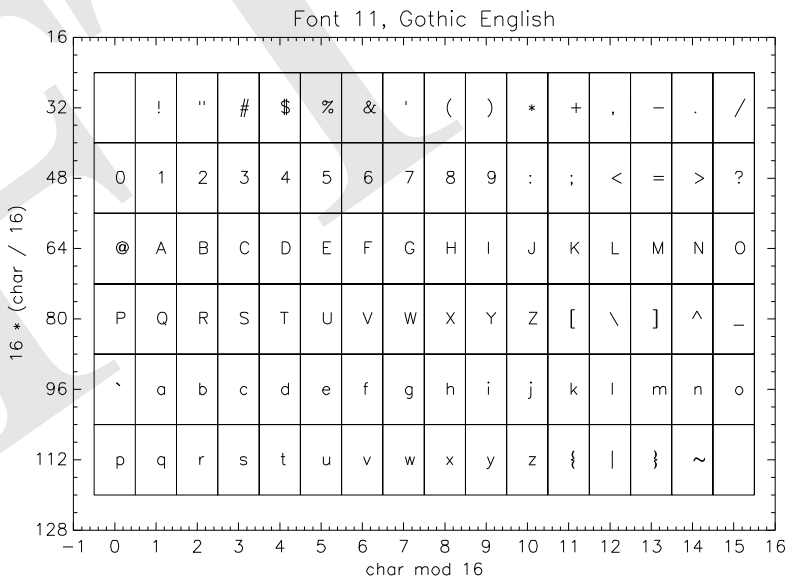
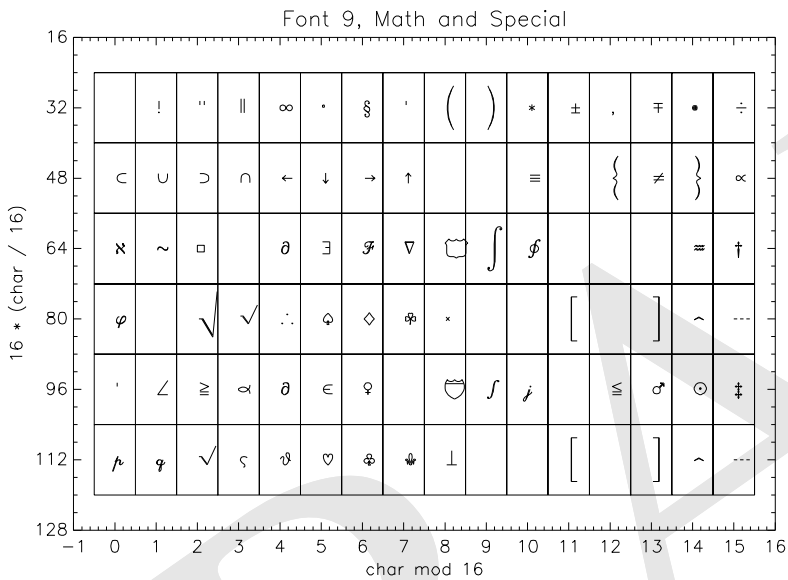


```
1 showfont , 8, 'Complex Italic '
```

```
1 showfont , 9, 'Math and Special '
```

% Compiled module: SHOWFONT.

% Compiled module: SHOWFONT.

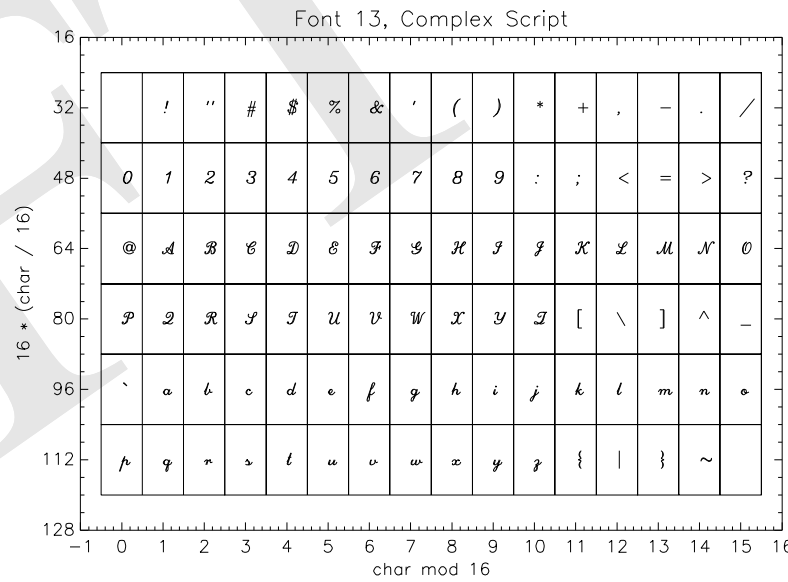
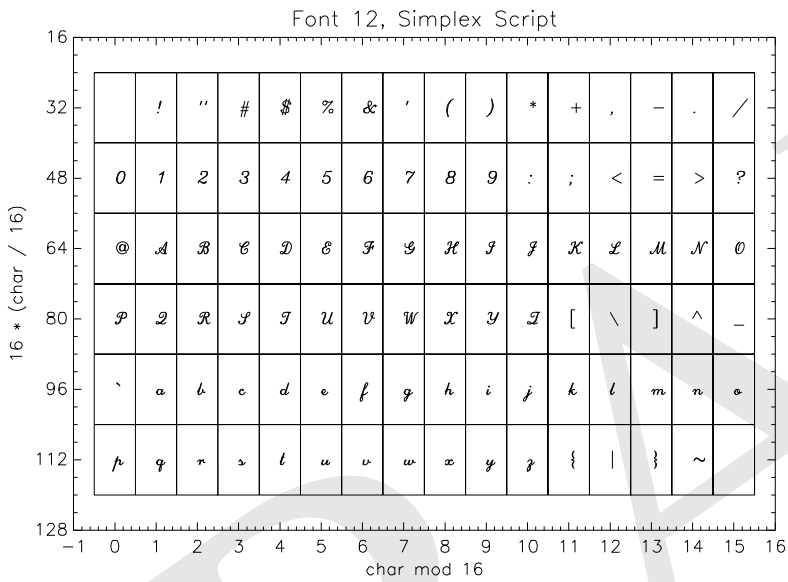


```
1 showfont , 11, 'Gothic English'
```

```
1 showfont , 12, 'Simplex Script'
```

```
% Compiled module: SHOWFONT.
```

```
% Compiled module: SHOWFONT.
```

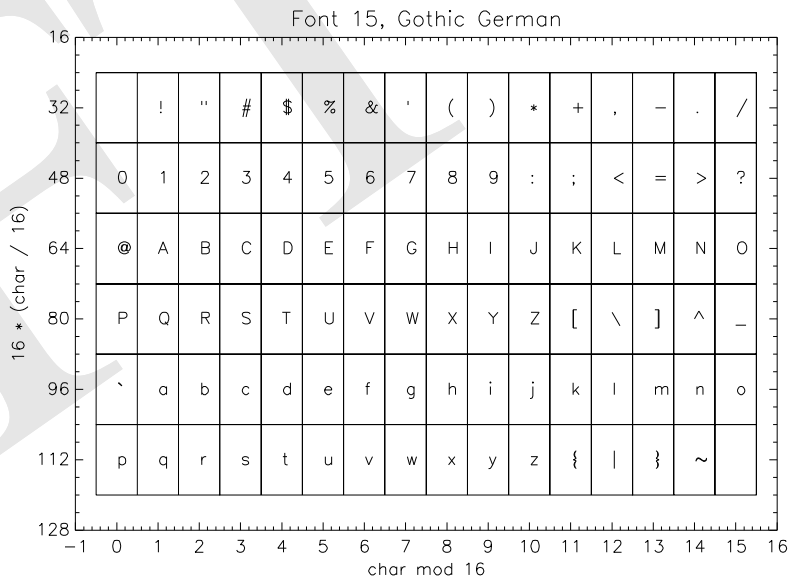
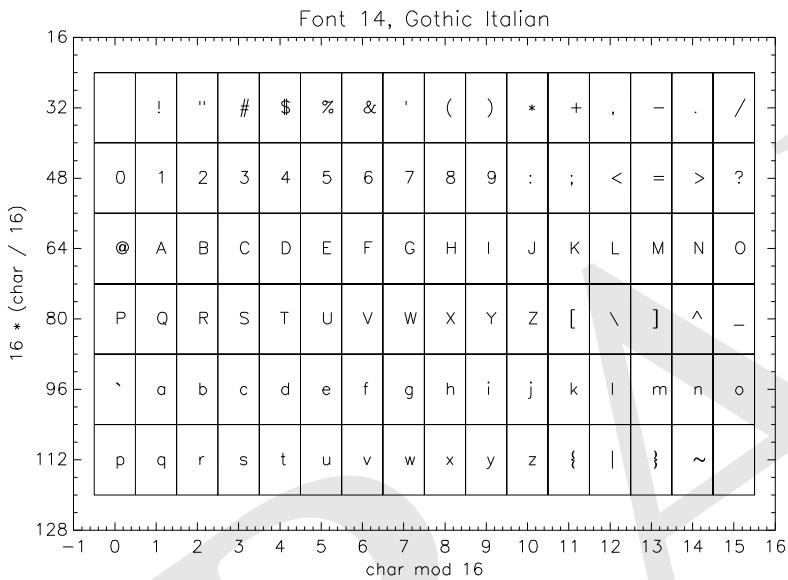


```
1 showfont , 13, 'Complex Script '
```

```
1 showfont , 14, 'Gothic Italian '
```

% Compiled module: SHOWFONT.

% Compiled module: SHOWFONT.

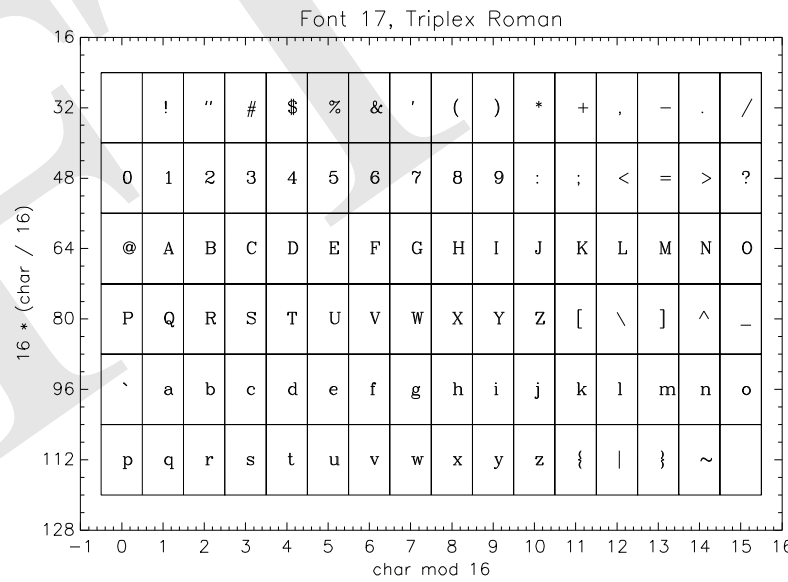
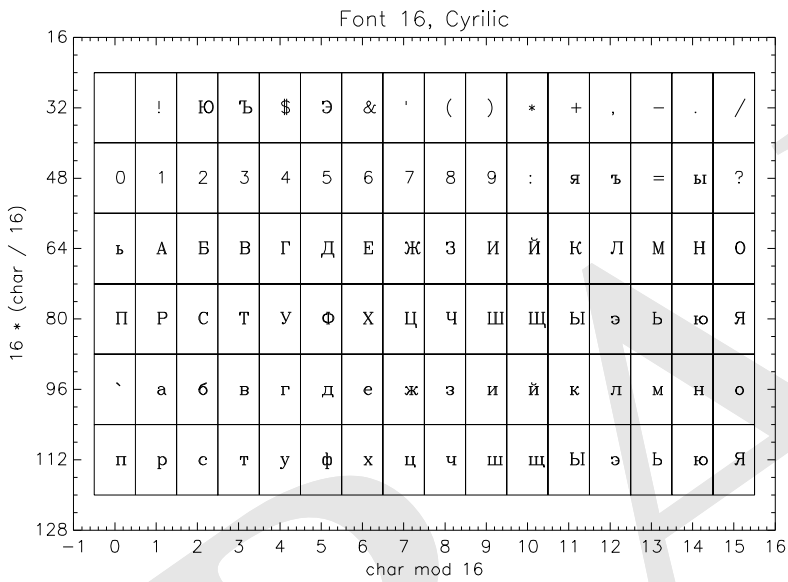


```
1 showfont , 15, 'Gothic German'
```

```
1 showfont , 16, 'Cyrilic'
```

% Compiled module: SHOWFONT.

% Compiled module: SHOWFONT.

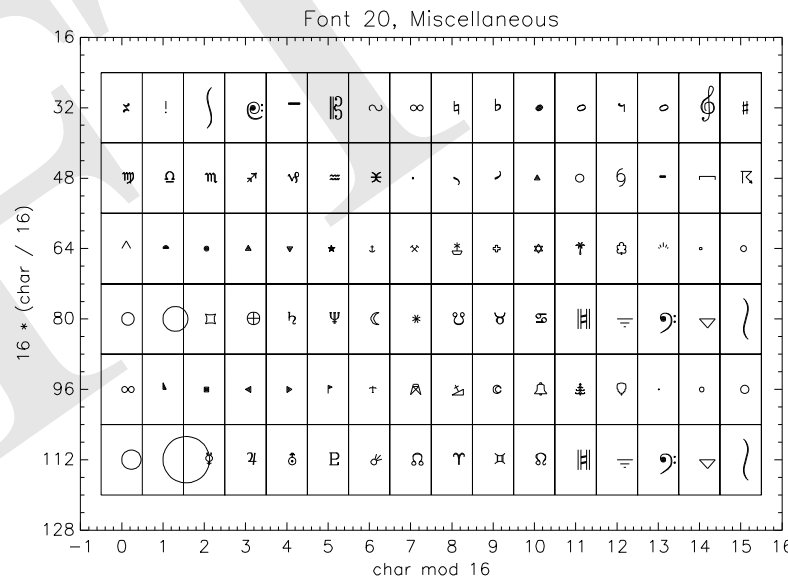
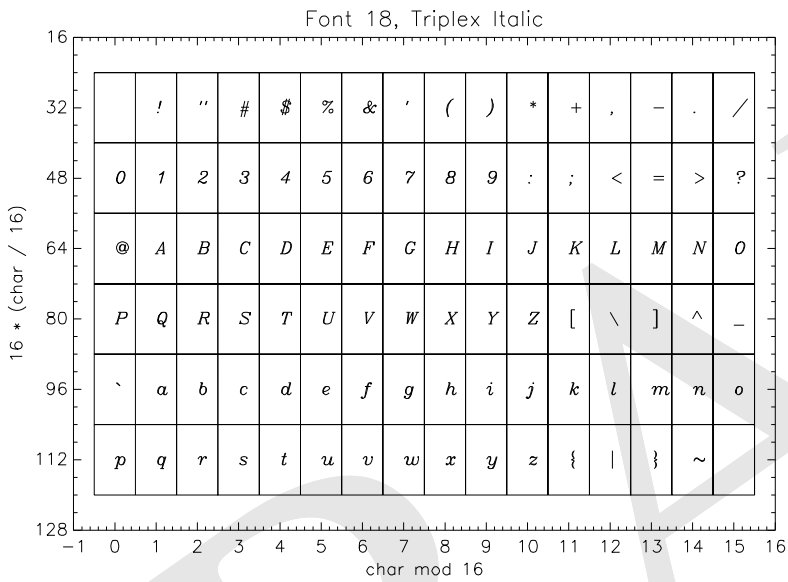


```
1 showfont , 17, 'Triplex Roman'
```

```
1 showfont , 18, 'Triplex Italic '
```

% Compiled module: SHOWFONT.

% Compiled module: SHOWFONT.



SIN() function

positional arguments: 1
keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

SINDGEN() function

positional arguments: 8
keywords: none

SINH() function

positional arguments: 1
keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

```
1 showfont , 20, 'Miscellaneous '
```

```
% Compiled module: SHOWFONT.
```

SIZE() function

positional arguments: 1

keywords: **DIMENSIONS**, **FILE_LUN**, **L64**, **N_DIMENSIONS**, **N_ELEMENTS**, **STRUCTURE**, **TNAME**, **TYPE**

SKEWNESS() function

positional arguments: 1

keywords: **DOUBLE**, **NAN**

SKIP_LUN procedure

positional arguments: 2

keywords: **EOF**, **HELP**, **LINES**, **TEST**, **TRANSFER_COUNT**

SMOOTH() function

positional arguments: 2

keywords: **EDGE_TRUNCATE**, **HELP**, **NAN**, **TEST**, **VERBOSE**

SOBEL() function

positional arguments: 1

keywords: **HELP**

SOCKET procedure

positional arguments: 3

keywords: **CONNECT_TIMEOUT**, **ERROR**, **GET_LUN**, **READ_TIMEOUT**, **STDIO**, **SWAP_ENDIAN**, **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**, **WIDTH**, **WRITE_TIMEOUT**

SORT() function

positional arguments: 1

keywords: **L64**

SPAWN procedure

positional arguments: 3

keywords: **COUNT**, **EXIT_STATUS**, **NOSHELL**, **PID**, **SH**, **UNIT**

SPHER_HARM() function

positional arguments: 4

keywords: **DOUBLE**

SPL_INIT() function

positional arguments: 2

keywords: **DOUBLE**, **HELP**, **YP0**, **YPN_1**

SPL_INIT_OLD() function

positional arguments: 2

keywords: **DEBUG**, **DOUBLE**, **YP0**, **YPN_1**

SPL_INTERP() function

positional arguments: 4

keywords: **DOUBLE**, **HELP**

SPL_INTERP_OLD() function

positional arguments: 4

keywords: **DOUBLE**

SQRT() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STDDEV() function

positional arguments: 1

keywords: DOUBLE, NAN

STOP procedure

positional arguments: any number

keywords: AM_PM, DAYS_OF_WEEK, FORMAT, MONTH, STUDIO_NON_FINITE

STRARR() function

positional arguments: 8

keywords: NOZERO

STRCMP() function

positional arguments: 3

keywords: FOLD_CASE

STRCOMPRESS() function

positional arguments: 1

keywords: REMOVE_ALL

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STREGEX() function

positional arguments: 2

keywords: BOOLEAN, EXTRACT, FOLD_CASE, LENGTH, SUBEXPR

STRING() function

positional arguments: any number

keywords: AM_PM, DAYS_OF_WEEK, FORMAT, MONTH, PRINT

PRINT keyword

```

1 help , string(55b)
2 help , string(55b, /print)
3 help , string(findgen(2,2))
4 help , string(findgen(2,2), /print)
5 help , string(findgen(2), /print)

```

<Expression>	STRING	=	'7'
<Expression>	STRING	=	' 55'
<Expression>	STRING	=	Array[2, 2]
<Expression>	STRING	=	Array[2]
<Expression>	STRING	=	' 0.00000 1.00000 '

STRJOIN() function

positional arguments: 2

keywords: SINGLE

```

1 arr = ['a', 'b', 'c']
2 str = strjoin(arr)
3 help , arr , str

```

ARR	STRING	=	Array[3]
STR	STRING	=	'abc'

```

1 arr = [['a', 'b', 'c'], ['d', 'e', 'f']]
2 str = strjoin(arr, '-')
3 help, arr, str
4 print, str[0]
5 print, str[1]

```

```

ARR          STRING    = Array[3, 2]
STR          STRING    = Array[2]
a-b-c
d-e-f

```

SINGLE keyword

```

1 arr = [['a', 'b', 'c'], ['d', 'e', 'f']]
2 str = strjoin(arr, '-', /single)
3 help, arr, str

```

```

ARR          STRING    = Array[3, 2]
STR          STRING    = 'a-b-c-d-e-f'

```

STRLEN() function

positional arguments: 1
keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRLOWCASE() function

positional arguments: 1
keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRMATCH() function

positional arguments: 2
keywords: FOLD_CASE

STRMID() function

positional arguments: 3
keywords: REVERSE_OFFSET

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRPOS() function

positional arguments: 3
keywords: REVERSE_OFFSET, REVERSE_SEARCH

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRPUT procedure

positional arguments: 3
keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRSPLIT() function

positional arguments: 2
keywords: COUNT, ESCAPE, EXTRACT, FOLD_CASE, HELP, LENGTH, PRESERVE_NULL, REGEX, TEST

STRTOK() function

positional arguments: 2
keywords: ESCAPE, EXTRACT, LENGTH, PRESERVE_NULL, REGEX

STRTRIM() function

positional arguments: 2

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STRUCT_ASSIGN procedure

positional arguments: 2

keywords: NOZERO, VERBOSE

STRUPCASE() function

positional arguments: 1

keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

STR_SEP() function

positional arguments: 2

keywords: ESC, HELP, REMOVE_ALL, TEST, TRIM

...STR_SEP separates the string on **any** of the characters of the 2nd string. ...

SURFACE procedure

positional arguments: 3

keywords: AX, AZ, BACKGROUND, CHARSIZE, CHARTHICK, CLIP, COLOR, DATA, DEVICE, FONT, LINESIZE, MAX_VALUE, MIN_VALUE, NO_CLIP, NODATA, NOERASE, NORMAL, POSITION, SUBTITLE, T3D, THICK, TICKLEN, TITLE, XCHARSIZE, XGRIDSTYLE, XLOG, XMARGIN, XMINOR, XRANGE, XSTYLE, XTHICK, XTICKFORMAT, XTICKINTERVAL, XTICKLAY-OUT, XTICKLEN, XTICKNAME, XTICKS, XTICKUNITS, XTICKV, XTICK_GET, XTITLE, XTYPE, YCHARSIZE, YGRIDSTYLE, YLOG, YMARGIN, YMINOR, YRANGE, YSTYLE, YTHICK, YTICKFORMAT, YTICKINTERVAL, YTICKLAY-OUT, YTICKLEN, YTICKNAME, YTICKS, YTICKUNITS, YTICKV, YTICK_GET,

YTITLE, YTYPE, ZCHARSIZE, ZGRIDSTYLE, ZLOG, ZMARGIN, ZMINOR, ZRANGE, ZSTYLE, ZTHICK, ZTICKFORMAT, ZTICKINTERVAL, ZTICKLAY-OUT, ZTICKLEN, ZTICKNAME, ZTICKS, ZTICKUNITS, ZTICKV, ZTICK_GET, ZTITLE, ZTYPE, ZVALUE

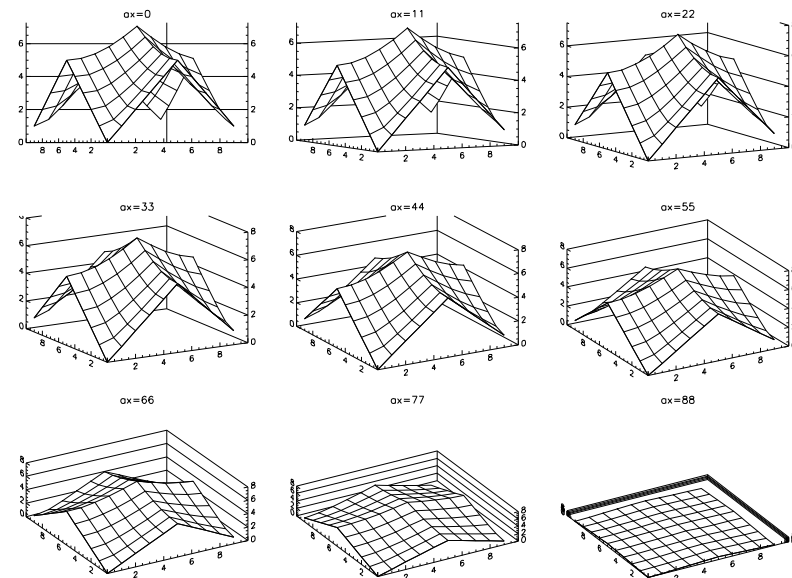
AX keyword

```

1 !P.MULTI = [0,3,3]
2 d = dist(10)
3 for ax = 0, 90, 11 do $
4   surface, d, ax=ax, title='ax=' + strtrim(ax,2)

```

% Compiled module: DIST.



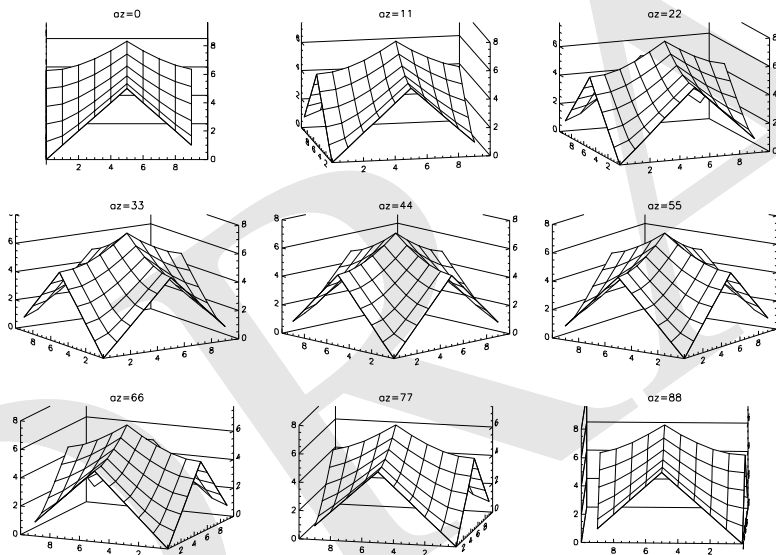
AZ keyword

```

1 !P.MULTI = [0,3,3]
2 d = dist(10)
3 for az = 0, 90, 11 do $
4   surface, d, az=az, title='az=' + strtrim(az,2)

```

```
% Compiled module: DIST.
```

**SVDC procedure**

positional arguments: 4
 keywords: **COLUMN**, **DOUBLE**, **ITMAX**

SWAP_ENDIAN() function

positional arguments: 1
 keywords: **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**

SWAP_ENDIAN_INPLACE procedure

positional arguments: 1
 keywords: **SWAP_IF_BIG_ENDIAN**, **SWAP_IF_LITTLE_ENDIAN**

SYSTIME() function

positional arguments: 2
 keywords: **JULIAN**, **SECONDS**, **UTC**

TAG_NAMES() function

positional arguments: 1
 keywords: **STRUCTURE_NAME**

TAN() function

positional arguments: 1
 keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

TANH() function

positional arguments: 1
 keywords: none

multi-threading: this routine uses GDL thread pool if working on large array, see the...

TEMPLATE procedure

positional arguments: none
 keywords: none

TEMPLATE_BLANK procedure

positional arguments: none
keywords: none

TEMPORARY() function

positional arguments: 1
keywords: none

TEST procedure

positional arguments: any number
keywords: none

TOTAL() function

positional arguments: 2
keywords: CUMULATIVE, DOUBLE, INTEGER, NAN, PRESERVE_TYPE

multi-threading: this routine uses GDL thread pool if working on large array, see the...

TRACE() function

positional arguments: 1
keywords: DOUBLE

TRANSPOSE() function

positional arguments: 2
keywords: none

TRIGRID() function

positional arguments: 6
keywords: MAP, MAX_VALUE, MISSING, NX, NY

TV procedure

positional arguments: 4
keywords: CHANNEL, DATA, DEVICE, NORMAL, ORDER, TRUE, XSIZE, YSIZE

TVLCT procedure

positional arguments: 4
keywords: GET, HLS, HSV

TVRD() function

positional arguments: 5
keywords: CHANNEL, ORDER, TRUE, WORDS

TVSCL procedure

positional arguments: 3
keywords: NAN, _EXTRA

T_PDF() function

positional arguments: 2
keywords: none

UINDGEN() function

positional arguments: 8
keywords: none

UINT() function

positional arguments: 10

keywords: none

UINTARR() function

positional arguments: 8

keywords: NOZERO

UL64INDGEN() function

positional arguments: 8

keywords: none

ULINDGEN() function

positional arguments: 8

keywords: none

ULON64ARR() function

positional arguments: 8

keywords: NOZERO

ULONARR() function

positional arguments: 8

keywords: NOZERO

ULONG() function

positional arguments: 10

keywords: none

ULONG64() function

positional arguments: 10

keywords: none

UNIQ() function

positional arguments: 2

keywords: none

USERSYM procedure

positional arguments: 2

keywords: COLOR, FILL, THICK

VALUE_LOCATE() function

positional arguments: 2

keywords: L64

VARIANCE() function

positional arguments: 1

keywords: DOUBLE, NAN

VOIGT() function

positional arguments: 2

keywords: DOUBLE, ITER

WAIT procedure

positional arguments: 1

keywords: none

WDELETE procedure

positional arguments: any number

keywords: none

WHERE() function

positional arguments: 2

keywords: COMPLEMENT, NCOMPLEMENT

see also: ARRAY_INDICES()

multi-threading: this routine uses GDL thread pool if working on large array, see the...

WIDGET_BASE() function

positional arguments: 1

keywords: ALIGN_BOTTOM, ALIGN_CENTER, ALIGN_LEFT, ALIGN_RIGHT, ALIGN_TOP, BASE_ALIGN_BOTTOM, BASE_ALIGN_CENTER, BASE_ALIGN_LEFT, BASE_ALIGN_RIGHT, BASE_ALIGN_TOP, COLUMN, CONTEXT_EVENTS, CONTEXT_MENU, DISPLAY_NAME, EVENT_FUNC, EVENT_PRO, EXCLUSIVE, FLOATING, FRAME, FUNC_GET_VALUE, GRID_LAYOUT, GROUP_LEADER, KBRD_FOCUS_EVENTS, KILL_NOTIFY, MAP, MBAR, MODAL, NONEXCLUSIVE, NOTIFY_REALIZE, NO_COPY, PRO_SET_VALUE, RESOURCE_NAME, RNAME_MBAR, ROW, SCROLL, SCR_XSIZE, SCR_YSIZE, SENSITIVE, SPACE, TITLE, TLB_FRAME_ATTR, TLB_ICONIFY_EVENTS, TLB_KILL_REQUEST_EVENTS, TLB_MOVE_EVENTS, TLB_SIZE_EVENTS, TOOLBAR, TRACKING_EVENTS, UNAME, UNITS, UVALUE, XOFFSET, XPAD, XSIZE, X_SCROLL_SIZE, YOFFSET, YPAD, YSIZE, Y_SCROLL_SIZE

WIDGET_BUTTON() function

positional arguments: 1

keywords: ACCELERATOR, ALIGN_CENTER, ALIGN_LEFT, ALIGN_RIGHT, BITMAP, CHECKED_MENU, DYNAMIC_RESIZE, EVENT_FUNC, EVENT_PRO, FONT, FRAME, FUNC_GET_VALUE, GROUP_LEADER, HELP, KILL_NOTIFY, MENU, NOTIFY_REALIZE, NO_COPY, NO_RELEASE, PRO_SET_VALUE, PUSHBUTTON_EVENTS, SCR_XSIZE, SCR_YSIZE, SENSITIVE, SEPARATOR,

TAB_MODE, TOOLTIP, TRACKING_EVENTS, UNAME, UNITS, UVALUE, VALUE, XOFFSET, XSIZE, X_BITMAP_EXTRA, YOFFSET, YSIZE

WIDGET_CONTROL procedure

positional arguments: 1

keywords: DESTROY, EVENT_PRO, FUNC_GET_VALUE, GET_UVALUE, GET_VALUE, MANAGED, MAP, NO_COPY, PRO_SET_VALUE, REALIZE, SENSITIVE, SET_BUTTON, SET_DROPLIST_SELECT, SET_UNAME, SET_UVALUE, SET_VALUE, XMANAGER_ACTIVE_COMMAND

WIDGET_DROPLIST() function

positional arguments: 1

keywords: DYNAMIC_RESIZE, EVENT_FUNC, EVENT_PRO, FONT, FRAME, FUNC_GET_VALUE, GROUP_LEADER, KILL_NOTIFY, NOTIFY_REALIZE, NO_COPY, PRO_SET_VALUE, RESOURCE_NAME, SCR_XSIZE, SCR_YSIZE, SENSITIVE, TAB_MODE, TITLE, TRACKING_EVENTS, UNAME, UNITS, UVALUE, VALUE, XOFFSET, XSIZE, YOFFSET, YSIZE

WIDGET_EVENT() function

positional arguments: 1

keywords: DESTROY, XMANAGER_BLOCK

WIDGET_INFO() function

positional arguments: 1

keywords: CHILD, MANAGED, MODAL, VALID, VERSION, XMANAGER_BLOCK

WIDGET_LABEL() function

positional arguments: 1

keywords: ALL_EVENTS, CONTEXT_EVENTS, EDITABLE, EVENT_FUNC, EVENT_PRO, FONT, FRAME, FUNC_GET_VALUE, GROUP_LEADER, IGNORE_ACCELERATORS, KBRD_FOCUS_EVENTS, KILL_NOTIFY, NOTIFY_REALIZE, NO_COPY, NO_NEWLINE, PRO_SET_VALUE, RESOURCE_NAME, SCROLL, SCR_XSIZE, SCR_YSIZE, SENSITIVE, TAB_MODE, TRACKING_EVENTS, UNAME, UNITS, UVALUE, VALUE, WRAP, XOFFSET, XSIZE, YOFFSET, YSIZE

WIDGET_TEXT() function

positional arguments: 1

keywords: ALL_EVENTS, CONTEXT_EVENTS, EDITABLE, EVENT_FUNC, EVENT_PRO, FONT, FRAME, FUNC_GET_VALUE, GROUP_LEADER, IGNORE_ACCELERATORS, KBRD_FOCUS_EVENTS, KILL_NOTIFY, NOTIFY_REALIZE, NO_COPY, NO_NEWLINE, PRO_SET_VALUE, RESOURCE_NAME, SCROLL, SCR_XSIZE, SCR_YSIZE, SENSITIVE, TAB_MODE, TRACKING_EVENTS, UNAME, UNITS, UVALUE, VALUE, WRAP, XOFFSET, XSIZE, YOFFSET, YSIZE

WINDOW procedure

positional arguments: 1

keywords: COLORS, FREE, PIXMAP, RETAIN, TITLE, XPOS, XSIZE, YPOS, YSIZE

WRITEU procedure

positional arguments: any number

keywords: TRANSFER_COUNT

WRITE_BMP procedure

positional arguments: 5

keywords: DEBUG, FOUR_BIT, HEADER_DEFINE, HELP, IHDR, RGB, TEST

WRITE_GIF procedure

positional arguments: 5

keywords: BACKGROUND_COLOR, CLOSE, DEBUG, DELAY_TIME, DISPOSAL_METHOD, HELP, MULTIPLE, REPEAT_COUNT, TEST, TRANSPARENT, USER_INPUT

WRITE_JPEG procedure

positional arguments: 2

keywords: DEBUG, HELP, ORDER, PROGRESSIVE, QUALITY, TEST, TRUE, UNIT

WRITE_PICT procedure

positional arguments: 5

keywords: DEBUG, HELP, TEST

WRITE_PNG procedure

positional arguments: 5

keywords: DEBUG, HELP, ORDER, TEST, TRANSPARENT, VERBOSE

WSET procedure

positional arguments: 1

keywords: none

WSHOW procedure

positional arguments: 2

keywords: none

WTN() function

positional arguments: 2

keywords: COLUMN, DOUBLE, INVERSE, OVERWRITE

XYOUTS procedure

positional arguments: 3

keywords: ALIGNMENT, CHARSIZE, CHARTHICK, CLIP, COLOR, DATA, DEVICE,
NOCLIP, NORMAL, ORIENTATION, WIDTH, Z

Part II

Developer's guide

Chapter 16

General remarks and coding guidelines

... such as the CERN C++ Coding Standard Specification [\[4\]](#) or other similar documents.

Chapter 17

The library-routine API

TODO: extract it using Doxygen or some similar tool.

Chapter 18

Extending the documentation

L^AT_EX

gdlldoc.sty

Natbib:

Chapter 19

Extending the testsuite (testsuite/README)

The list of GDL routines to be executed during the make-check run is defined in the testsuite/Makefile.am file. After adding a new item (filename) to the list, please rerun "automake" being in the root folder of the source tree. CMake also uses the list in Makefile.am.

Each test routine is invoked using the GDL "-e" command-line option by the "try" shell script in the testsuite directory (and in an analogous manner for the case of CMake/CTest). "make" decides on the status of a test basing on the exit code of this script:

- "success" for exit code 0
 - "ignorable failure" for code 77
 - "failure" for any other exit code, e.g. 1
- The "try" script should, in principle, exit with the GDL exit code. Therefore, a failure of a GDL test should be indicated by e.g.:

```
if ( ...true if test failed... ) begin
  message, 'reason for the failure', /continue
  exit, status=1
endif
```

An ignorable failure can be indicated by e.g.:

```
if (!XXX_exists()) then begin
  message, 'GDL was built w/o support for XXX - skipping', /conti
  exit, status=77
endif
```

Any GDL error (e.g. parser error or library-routine-triggered error) causing GDL to return to the \$MAIN\$ level will cause make to assume

success! (GDL exits normally in this case). Any GDL error causing GDL to stop execution on an other-than-\$MAIN\$ level will bring the GDL interpreter prompt.

The name of the file must match the name of the test routine, e.g. testsuite/test_dummy.pro for

```
pro test_dummy
...
end
```

GDL segfaults, assertion-exits, std::terminate() exits, etc. are handled as failures by make.

The "try" script always uses the gdl binary in the build tree - not the one installed in the system. The "try" script also sets appropriate env. variables so that the GDL-written library routines are taken from the source tree as well (e.g. src/pro/mean.pro).

Regardless if the autotools or the CMake/CTest configuration mechanism, the testsuite run is invoked by "make check" (not the default CMakes's "make test").

Chapter 20

A short overview of how GDL works internally

Programs (*.pro files) or command line input is parsed (GDLLexer.cpp, GDLParser.cpp generated with ANTLR from gdlc.g). These results in an abstract syntax tree (AST) consisting of 'DNode' (dnode.hpp). This syntax tree is further manipulated (compiled) with a tree parser (GDLTreeParser.cpp generated with ANTLR from gdlc.tree.g, dcompiler.hpp). Here the AST is splitted into the different functions/procedures and the DNode(s) are annotated with further information and converted to ProgNode(s). Then these compiled (ProgNode) ASTs are interpreted (GDLInterpreter.cpp generated with ANTLR from gdlc.i.g, dinterpreter.cpp).

Chapter 21

How to make use of OpenMP in GDL

Chapter 22

Notes for packagers

Optional features of PLplot and ImageMagick

The HDF4-netCDF conflict

Part III

Indices

Subject Index

- .COMPILE, 17
- .CONTINUE, 17
- .STEP, 17
- \$MAIN\$, 73
- _EXTRA, 14
- _REF_EXTRA, 14
- _STRICT_EXTRA, 14
- _EXTRA keyword
 - in ISHFT() function, 56
 - in PLOTERR procedure, 67
 - in TVSCL procedure, 89
- _REF_EXTRA keyword
 - in CALL_FUNCTION() function, 38
 - in CALL_METHOD procedure, 38
 - in CALL_METHOD() function, 38
 - in CALL_PROCEDURE procedure, 38
 - in OBJ_DESTROY procedure, 65
 - in OBJ_NEW() function, 65
 - in QUERY_IMAGE() function, 70
- abbreviated keyword names, 14
- ABORT keyword
 - in NCDF_CONTROL procedure, 63
- ABS() function, 18, 33
- ACCELERATOR keyword
 - in WIDGET_BUTTON() function, 91
- ACOS() function, 19, 33
- ALIGN_BOTTOM keyword
 - in WIDGET_BASE() function, 91
- ALIGN_CENTER keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
- ALIGN_LEFT keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
- ALIGN_RIGHT keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
- ALIGN_TOP keyword
 - in WIDGET_BASE() function, 91
- ALIGNMENT keyword
 - in XYOUTS procedure, 93
- ALL_DIRS keyword
 - in EXPAND_PATH() function, 43
- ALL_EVENTS keyword
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- ALL_GDL keyword
 - in CALL_EXTERNAL() function, 36
- ALL_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- ALL keyword
 - in CLOSE procedure, 39
 - in HDF_OPEN() function, 50
 - in SAVE procedure, 75
- ALLOCATE_HEAP keyword
 - in PTR_NEW() function, 69
 - in PTRARR() function, 68
- ALLOW_NONEXISTENT keyword
 - in FILE_DELETE procedure, 44
- ALLOW_SAME keyword
 - in FILE_COPY procedure, 44
- ALOG() function, 18, 33
- ALOG10() function, 18, 33
- AM_PM keyword
 - in PRINT procedure, 68
 - in PRINTF procedure, 68
 - in READ procedure, 70
 - in READF procedure, 70
 - in READS procedure, 70
 - in STOP procedure, 85
 - in STRING() function, 85
- APPEND keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SAVE procedure, 75
- APPLEMAN procedure, 33
- ARG_NAME keyword
 - in ROUTINE_NAMES() function, 73
- ARG_PRESENT() function, 14, 34, 74
- ARGV keyword
 - in PYTHON procedure, 69
 - in PYTHON() function, 69
- ARRAY_EQUAL() function, 15, 34
- ARRAY_INDICES() function, 15, 34, 52, 91
- ARRAY keyword
 - in EXPAND_PATH() function, 43
- ASIN() function, 19, 34
- ASSOC() function, 12, 34
- ATAN() function, 12, 19, 33, 34
- ATRANSPOSE keyword
 - in MATRIX_MULTIPLY() function, 61
- AX keyword
 - in SURFACE procedure, 87
- AXIS procedure, 35

AXISprocedure, 23
 AZ keyword
 in SURFACE procedure, 87

B_VALUE keyword
 in CALL_EXTERNAL() function, 36

BACKGROUND_COLOR keyword
 in WRITE_GIF procedure, 92

BACKGROUND keyword
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

BACKPROJECT keyword
 in RADON() function, 70

BAR_COLOR keyword
 in PLOTERR procedure, 67

BASE_ALIGN_BOTTOM keyword
 in WIDGET_BASE() function, 91

BASE_ALIGN_CENTER keyword
 in WIDGET_BASE() function, 91

BASE_ALIGN_LEFT keyword
 in WIDGET_BASE() function, 91

BASE_ALIGN_RIGHT keyword
 in WIDGET_BASE() function, 91

BASE_ALIGN_TOP keyword
 in WIDGET_BASE() function, 91

BASE keyword
 in SHOWFONT procedure, 75

BEG keyword
 in SHOWFONT procedure, 75

BEGIN, 14
 in CASE statement, 13
 in FOR statement, 13
 in IF/THEN/ELSE statement, 12
 in SWITCH statement, 13
 in WHILE statement, 14

BESELI() function, 19, 35

BESELJ() function, 19, 35

BESELK() function, 19, 35

BESELY() function, 19, 35

BETA() function, 19, 35

BILINEAR() function, 35

BIN1 keyword
 in HIST_2D() function, 52

BIN2 keyword
 in HIST_2D() function, 52

BINARY keyword
 in OPENR procedure, 66
 in OPENU procedure, 66

 in OPENW procedure, 66

BINDGEN() function, 13, 15, 35

BINOMIAL keyword
 in RANDOMN() function, 70
 in RANDOMU() function, 70

BINSIZE keyword
 in HISTOGRAM() function, 52

BITMAP keyword
 in WIDGET_BUTTON() function, 91

BLOCK_SPECIAL keyword
 in FILE_TEST() function, 45

BLOCK keyword
 in OPENR procedure, 66
 in OPENU procedure, 66

 in OPENW procedure, 66

BOOLEAN keyword
 in STREGEX() function, 85

BOTTOM keyword
 in LOADCT procedure, 57

BREAK
 in CASE statement, 13
 in FOR statement, 13
 in FOREACH statement, 13
 in REPEAT statement, 14
 in SWITCH statement, 13
 in WHILE statement, 14

BREAKDOWN_EPOCH keyword
 in CDF_EPOCH procedure, 38

BRIEF keyword
 in HELP procedure, 52

BROYDEN() function, 20, 35

BTRANPOSE keyword
 in MATRIX_MULTIPLY() function, 61

BUFFER keyword
 in READ_JPEG procedure, 71

BUFSIZE keyword
 in OPENR procedure, 66
 in OPENU procedure, 66

 in OPENW procedure, 66

BYTARR() function, 13, 15, 35

BYTE() function, 12, 13, 35

BYTE keyword
 in HDF_SD_CREATE() function, 50
 in INDGEN() function, 56

 in MAKE_ARRAY() function, 61
 in NCDF_ATTPUT procedure, 62

 in NCDF_VARDEF() function, 63

BYTEORDER procedure, 36

BYTEORDERprocedure, 19, 21

BYTSCL() function, 23, 28, 36

C_CHARSIZE keyword
 in CONTOUR procedure, 39

C_COLORS keyword
 in CONTOUR procedure, 39

C_LINestyle keyword
 in CONTOUR procedure, 39

cal (UNIX), 36

CALDAT procedure, 36

CALDATprocedure, 27

CALENDAR procedure, 36

CALENDARprocedure, 27

CALL_EXTERNAL() function, 31, 36, 57

CALL_FUNCTION() function, 14, 38

CALL_METHOD procedure, 38

CALL_METHOD() function, 14, 38

CALL_METHODprocedure, 14

CALL_METHON() function, 15

CALL_METHONprocedure, 15

CALL_PROCEDURE procedure, 38

CALL_PROCEDURE() function, 14

CALLS keyword
 in HELP procedure, 52

CANCEL keyword

- in CATCH procedure, 38
 - in DIALOG_MESSAGE() function, 42
- CASE, 13
- CAST keyword
 - in OBJ_VALID() function, 66
 - in PTR_VALID() function, 69
- CATCH procedure, 38
- CATCHprocedure, 15
- CD procedure, 38
- CDF_EPOCH procedure, 38
- CDprocedure, 25
- CEIL() function, 18, 39
- CENTER keyword
 - in CONGRID() function, 39
 - in CONVOL() function, 40
 - in DIALOG_MESSAGE() function, 42
- CHANGE keyword
 - in CURSOR procedure, 41
- CHANNEL keyword
 - in TV procedure, 89
 - in TVRD() function, 89
- CHANNELS keyword
 - in MAGICK_PING() function, 60
 - in READ_TIFF() function, 71
- CHAR keyword
 - in NCDF_ATTPUT procedure, 62
 - in NCDF_VARDEF() function, 63
- CHARACTER_SPECIAL keyword
 - in FILE_TEST() function, 45
- CHARSIZE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
 - in XYOUTS procedure, 93
- CHARTHICK keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
 - in XYOUTS procedure, 93
- CHECK_MATH() function, 15, 39
- CHECK_MATHprocedure, 17
- CHECKED_MENU keyword
 - in WIDGET_BUTTON() function, 91
- CHILD keyword
 - in WIDGET_INFO() function, 91
- CINDGEN() function, 13, 15, 39
- CLASS keyword
 - in HDF_VD_GET procedure, 51
 - in HDF_VG_GETINFO procedure, 51
- CLIENTSERVER keyword
 - in LMGR() function, 57
- CLIP_PLANE keyword
 - in MAP_CLIP_SET procedure, 61
- CLIP_UV keyword
 - in MAP_CLIP_SET procedure, 61
- CLIP keyword
 - in CONTOUR procedure, 39
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SURFACE procedure, 87
 - in XYOUTS procedure, 93
- CLOBBER keyword
 - in NCDF_CREATE() function, 63
- CLOSE procedure, 39
- CLOSE_FILE keyword
 - in DEVICE procedure, 41
- CLOSE keyword
 - in WRITE_GIF procedure, 92
- CLOSEprocedure, 21
- COEFFICIENTS keyword
 - in LAGUERRE() function, 56
- COLOR keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in DEVICE procedure, 41
 - in MAP_CONTINENTS procedure, 61
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
- in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SURFACE procedure, 87
 - in USERSYM procedure, 90
 - in XYOUTS procedure, 93
- COLORS keyword
 - in READ_JPEG procedure, 71
 - in WINDOW procedure, 92
- COLUMN keyword
 - in LUDC procedure, 59
 - in LUSOL() function, 59
 - in SVDC procedure, 88
 - in WIDGET_BASE() function, 91
 - in WTN() function, 93
- COMMAND_LINE_ARGS() function, 25, 39
- COMMENT_SYMBOL keyword
 - in READ_ASCII() function, 71
- COMPANION keyword
 - in IMSL_ZEROPOLY() function, 55
- COMPATIBLE keyword
 - in SAVE procedure, 75
- COMPLEMENT keyword
 - in WHERE() function, 91
- complex numbers
 - magnitude, 18
- COMPLEX() function, 12, 13, 39
- COMPLEX keyword
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
- COMPLEXARR() function, 13, 15, 39
- COMPRESS keyword
 - in FILE_LINES() function, 45
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
- COMPUTE_EPOCH keyword
 - in CDF_EPOCH procedure, 38
- CONGRID() function, 39
- CONJ() function, 12, 39
- CONNECT_TIMEOUT keyword
 - in SOCKET procedure, 84

- CONTEXT_EVENTS keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- CONTEXT_MENU keyword
 - in WIDGET_BASE() function, 91
- CONTINUE
 - in CONTINUE statement, 14
 - in FOR statement, 13
 - in FOREACH statement, 13
 - in WHILE statement, 14
- CONTINUE keyword
 - in MESSAGE procedure, 62
 - in PLOTS procedure, 67
- CONTOUR procedure, 39
- CONTOURprocedure, 23
- CONVERT_ALL keyword
 - in IDL_VALIDNAME() function, 53
- CONVERT_COORD() function, 24, 40
- CONVERT_SPACES keyword
 - in IDL_VALIDNAME() function, 53
- CONVOL() function, 18, 28, 40
- COORDSYS keyword
 - in HDF_SD_GETINFO procedure, 51
- COPY keyword
 - in SET_PLOT procedure, 75
- CORRELATE() function, 19, 40
- COS() function, 19, 40
- COSH() function, 19, 41
- COUNT keyword
 - in COMMAND_LINE_ARGS() function, 39
 - in EXPAND_PATH() function, 43
 - in FILE_SEARCH() function, 45
 - in FINDFILE() function, 45
 - in GET_DRIVE_LIST() function, 47
 - in HDF_SD_ADDDATA procedure, 50
 - in HDF_SD_ATTRINFO procedure, 50
 - in HDF_SD_DIMGET procedure, 50
 - in HDF_SD_GETDATA procedure, 51
 - in HDF_VD_GET procedure, 51
 - in IMAGE_STATISTICS procedure, 53
 - in NCDF_VARGET procedure, 63
 - in NCDF_VARPUT procedure, 64
 - in OBJ_CLASS() function, 64
 - in OBJ_VALID() function, 66
 - in PTR_VALID() function, 69
 - in READ_ASCII() function, 71
 - in SPAWN procedure, 84
 - in STRSPLIT() function, 86
- COUNTRIES keyword
 - in MAP_CONTINENTS procedure, 61
- COVARIANCE keyword
 - in CORRELATE() function, 40
- CPU procedure, 41
- CPUprocedure, 29
- CREATE_STRUCT() function, 15, 41
- CREATE keyword
 - in HDF_OPEN() function, 50
 - in HDF_SD_START() function, 51
- CROSSP() function, 18, 41
- CUBIC keyword
 - in CONGRID() function, 39
 - in INTERPOLATE() function, 56
 - in POLY_2D() function, 68
- CUMULATIVE keyword
 - in PRODUCT() function, 68
 - in TOTAL() function, 89
- CURRENT keyword
 - in CD procedure, 38
 - in MEMORY() function, 62
- CURSOR procedure, 41
- CURSORprocedure, 23
- D_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- DATA_DIMS keyword
 - in READ_BINARY() function, 71
- DATA_LENGTH keyword
 - in N_TAGS() function, 64
- DATA_START keyword
 - in READ_ASCII() function, 71
 - in READ_BINARY() function, 71
- DATA_SUM keyword
 - in IMAGE_STATISTICS procedure, 53
- DATA_TYPE keyword
 - in READ_BINARY() function, 71
- DATA keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in CONVERT_COORD() function, 40
 - in CURSOR procedure, 41
 - in HDF_SD_ATTRINFO procedure, 50
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SAVE procedure, 75
 - in SURFACE procedure, 87
 - in TV procedure, 89
 - in XYOUTS procedure, 93
- DAYS_OF_WEEK keyword
 - in PRINT procedure, 68
 - in PRINTF procedure, 68
 - in READ procedure, 70
 - in READF procedure, 70
 - in READS procedure, 70
 - in STOP procedure, 85
 - in STRING() function, 85
- DBLARR() function, 13, 15, 41
- DCINDGEN() function, 13, 15, 41
- DCOMPLEX() function, 12, 13, 41
- DCOMPLEX keyword
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
- DCOMPLEXARR() function, 13, 15, 41
- DEBUG keyword
 - in DIALOG_PICKFILE() function, 42
 - in FILE_WHICH() function, 45
 - in READ_GIF procedure, 71
 - in READ_JPEG procedure, 71
 - in SPL_INIT_OLD() function, 84
 - in WRITE_BMP procedure, 92
 - in WRITE_GIF procedure, 92
 - in WRITE_JPEG procedure, 92

- in IDENTITY() function, 53
- in IGAMMA() function, 53
- in IMSL_BINOMIALCOEF() function, 53
- in IMSL_CONSTANT() function, 54
- in IMSL_ERF() function, 55
- in IMSL_ZEROPOLY() function, 55
- in IMSL_ZEROSYS() function, 56
- in INDGEN() function, 56
- in INVERT() function, 56
- in KURTOSIS() function, 56
- in LA_TRIRED procedure, 56
- in LAGUERRE() function, 56
- in LEGENDRE() function, 57
- in LNGAMMA() function, 57
- in LUDC procedure, 59
- in LUSOL() function, 59
- in MACHAR() function, 59
- in MAKE_ARRAY() function, 61
- in MEAN() function, 62
- in MEANABSDEV() function, 62
- in MEDIAN() function, 62
- in MOMENT() function, 62
- in NCDF_ATTPUT procedure, 62
- in NCDF_VARDEF() function, 63
- in NEWTON() function, 64
- in NORM() function, 64
- in POLY_AREA() function, 68
- in RADON() function, 70
- in RANDOMN() function, 70
- in RANDOMU() function, 70
- in RK4() function, 72
- in RK4JMG() function, 72
- in SKEWNESS() function, 84
- in SPHER_HARM() function, 84
- in SPL_INIT() function, 84
- in SPL_INIT_OLD() function, 84
- in SPL_INTERP() function, 84
- in SPL_INTERP_OLD() function, 84
- in STDDEV() function, 85
- in SVDC procedure, 88
- in TOTAL() function, 89

- in TRACE() function, 89
- in VARIANCE() function, 90
- in VOIGT() function, 90
- in WTN() function, 93
- DOWN keyword
 - in CURSOR procedure, 41
- DRHO keyword
 - in RADON() function, 70
- DTOXDR keyword
 - in BYTEORDER procedure, 36
- DX keyword
 - in RADON() function, 70
- DY keyword
 - in RADON() function, 70
- DYNAMIC_RESIZE keyword
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
- EDGE_TRUNCATE keyword
 - in CONVOL() function, 40
 - in SMOOTH() function, 84
- EDGE_WRAP keyword
 - in CONVOL() function, 40
- EDITABLE keyword
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- ELSE
 - in CASE statement, 13
 - in IF/THEN/ELSE statement, 12
 - in SWITCH statement, 13
- EMBEDDED keyword
 - in LMGR() function, 57
- ENABLED keyword
 - in ROUTINE_INFO() function, 72
- ENCAPSULATED keyword
 - in DEVICE procedure, 41
 - in SHOWFONT procedure, 75
- ENDCASE, 13
- ENDEF keyword
 - in NCDF_CONTROL procedure, 63
- ENDELSE, 12

- ENDFOR, 13
- ENDFOREACH, 13
- ENDIAN keyword
 - in READ_BINARY() function, 71
- ENDIF, 12
- ENDREP, 14
- ENDSWITCH, 13
- ENDWHILE, 14
- ENVIRONMENT keyword
 - in GETENV() function, 47
- EOF() function, 42
- EOF keyword
 - in SKIP_LUN procedure, 84
- EOFprocedure, 21
- ERASE procedure, 42
- ERASEprocedure, 23
- ERF() function, 18, 42
- ERFC() function, 18, 43
- ERR_REL keyword
 - in IMSL_ZEROSYS() function, 56
- ERRMSG keyword
 - in SAVE procedure, 75
- ERROR keyword
 - in DIALOG_MESSAGE() function, 42
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
- ERRORF() function, 18, 43
- ESC keyword
 - in STR_SEP() function, 87
- ESCAPE_SPECIAL_CHAR() function, 43
- ESCAPE keyword
 - in STRSPLIT() function, 86
 - in STRTOK() function, 86
- EVEN keyword
 - in MEDIAN() function, 62
- EVENT_FUNC keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91

- in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- EVENT_PRO keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_CONTROL procedure, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- EXCLUSIVE keyword
 - in WIDGET_BASE() function, 91
- EXECUTABLE keyword
 - in FILE_TEST() function, 45
- EXECUTE() function, 14, 25, 43
- EXECUTEprocedure, 15
- EXISTS keyword
 - in DEFSYSV procedure, 41
- EXIT procedure, 43
- EXIT_STATUS keyword
 - in CLOSE procedure, 39
 - in FREE_LUN procedure, 46
 - in SPAWN procedure, 84
- EXITprocedure, 25
- EXP() function, 18, 43
- EXPAND_ENVIRONMENT keyword
 - in FILE_SEARCH() function, 45
- EXPAND_PATH() function, 14, 25, 43
- EXPAND_TILDE keyword
 - in FILE_SEARCH() function, 45
- EXPINT() function, 18, 43
- EXPIRE_DATE keyword
 - in LMGR() function, 57
- EXTRACT keyword
 - in STREGEX() function, 85
 - in STRSPLIT() function, 86
 - in STRTOK() function, 86
- F77_UNFORMATTED keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
- F_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- FACTORIAL() function, 19, 43
- FETCH keyword
 - in ROUTINE_NAMES() function, 73
- FFT() function, 20, 44
- FIELDS keyword
 - in HDF_VD_READ() function, 51
- FILE_BASENAME() function, 25, 44
- FILE_COPY procedure, 44
- FILE_COPYprocedure, 25
- FILE_DELETE procedure, 44
- FILE_DELETEprocedure, 25
- FILE_DIRNAME() function, 25, 44, 45
- FILE_EXPAND_PATH() function, 25, 45
- FILE_INFO() function, 25, 45
- FILE_LINES() function, 25, 45
- FILE_LUN keyword
 - in SIZE() function, 84
- FILE_MKDIR procedure, 45
- FILE_MKDIRprocedure, 25
- FILE_SAME() function, 25, 45
- FILE_SEARCH() function, 25, 45
- FILE_TEST() function, 25, 45
- FILE_WHICH() function, 25, 45
- FILE keyword
 - in CLOSE procedure, 39
 - in DIALOG_PICKFILE() function, 42
 - in LOADCT procedure, 57
- FILENAME keyword
 - in DEVICE procedure, 41
 - in RESTORE procedure, 72
 - in SAVE procedure, 75
- FILEPATH() function, 14, 44
- FILL_CONTINENTS keyword
 - in MAP_CONTINENTS procedure, 61
- FILL keyword
 - in CONTOUR procedure, 39
 - in NCDF_CONTROL procedure, 63
 - in USERSYM procedure, 90
- FILTER keyword
 - in DIALOG_PICKFILE() function, 42
- FIN keyword
 - in SHOWFONT procedure, 75
- FINDEX() function, 19, 45
- FINDFILE() function, 25, 45
- FINDGEN() function, 13, 15, 46
- FINITE() function, 15, 46
- FIX() function, 12, 13, 46
- FIX_FILTER keyword
 - in DIALOG_PICKFILE() function, 42
- FLOAT() function, 12, 13, 46
- FLOAT keyword
 - in HDF_SD_CREATE() function, 50
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
 - in NCDF_ATTPUT procedure, 62
 - in NCDF_VARDEF() function, 63
- FLOATING keyword
 - in WIDGET_BASE() function, 91
- FLOOR() function, 18, 46
- FLTARR() function, 13, 15, 46
- FLUSH procedure, 46
- FLUSHprocedure, 23
- FNORM keyword
 - in IMSL_ZEROSYS() function, 56
- FOLD_CASE keyword
 - in FILE_BASENAME() function, 44
 - in FILE_SEARCH() function, 45
 - in STRCMP() function, 85
 - in STREGEX() function, 85
 - in STRMATCH() function, 86
 - in STRSPLIT() function, 86
- FOLLOW keyword
 - in CONTOUR procedure, 39
- FONT keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92

- in WIDGET_TEXT() function, 92
- FOR, 13
- FORCE_DEMO keyword
 - in LMGR() function, 57
- FORCE keyword
 - in CLOSE procedure, 39
 - in FREE_LUN procedure, 46
- FOREACH, 13
- FORMAT keyword
 - in HDF_SD_GETINFO procedure, 51
 - in PM procedure, 67
 - in PRINT procedure, 68
 - in PRINTF procedure, 68
 - in READ procedure, 70
 - in READF procedure, 70
 - in READS procedure, 70
 - in STOP procedure, 85
 - in STRING() function, 85
- FOUR_BIT keyword
 - in WRITE_BMP procedure, 92
- FRAME keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- FREE_LUN procedure, 46
- FREE_LUNprocedure, 21
- FREE keyword
 - in WINDOW procedure, 92
- FSTAT() function, 25, 46
- FTOXDR keyword
 - in BYTEORDER procedure, 36
- FULL_INTERLACE keyword
 - in HDF_VD_READ() function, 51
- FULL_STRUCT keyword
 - in HELPFORM() function, 52
- FULLY_QUALIFY_PATH keyword
 - in FILE_SEARCH() function, 45
- FUNC_GET_VALUE keyword
 - in WIDGET_BASE() function, 91

- in WIDGET_BUTTON() function, 91
- in WIDGET_CONTROL procedure, 91
- in WIDGET_DROPLIST() function, 91
- in WIDGET_LABEL() function, 92
- in WIDGET_TEXT() function, 92
- FUNCTIONS keyword
 - in HELP procedure, 52
 - in ROUTINE_INFO() function, 72
- GAMMA() function, 19, 46
- GAMMA keyword
 - in RANDOMN() function, 70
 - in RANDOMU() function, 70
- Gauss symbol, 18
- GAUSS_CVF() function, 19, 46
- GAUSS_PDF() function, 19, 46
- Gaussian probability function, 19
- GAUSSIANNNOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- GAUSSINT() function, 19, 46
- GDL_ERFINV() function, 47
- GEOTIFF keyword
 - in READ_TIFF() function, 71
- GET_DECOMPOSED keyword
 - in DEVICE procedure, 41
- GET_DRIVE_LIST() function, 47
- GET_KBRD() function, 47
- GET_KBRDprocedure, 21
- GET_LOGIN_INFO() function, 47
- GET_LUN procedure, 47
- GET_LUN keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
- GET_LUNprocedure, 21
- GET_MODE keyword
 - in FILE_TEST() function, 45
- GET_NAMES keyword
 - in LOADCT procedure, 57

- in LOADCT_INTERNALGDL procedure, 58
- GET_PATH keyword
 - in DIALOG_PICKFILE() function, 42
- GET_SCREEN_SIZE() function, 24, 47
- GET_SCREEN_SIZE keyword
 - in DEVICE procedure, 41
- GET_UVALUE keyword
 - in WIDGET_CONTROL procedure, 91
- GET_VALUE keyword
 - in WIDGET_CONTROL procedure, 91
- GET_VISUAL_DEPTH keyword
 - in DEVICE procedure, 41
- GET keyword
 - in TVLCT procedure, 89
- GETENV() function, 25, 47
- GLOBAL keyword
 - in NCDF_ATTDEL procedure, 62
 - in NCDF_ATTGET procedure, 62
 - in NCDF_ATTINQ() function, 62
 - in NCDF_ATTNAME() function, 62
 - in NCDF_ATTPUT procedure, 62
 - in NCDF_ATTRENAME procedure, 63
- GOTO statement, 14
- GRAY keyword
 - in RADON() function, 70
- GRAYSCALE keyword
 - in MAGICK_QUANTIZE procedure, 61
 - in READ_JPEG procedure, 71
- GRIBAPI_CLONE() function, 22, 47
- GRIBAPI_CLOSE_FILE procedure, 47
- GRIBAPI_CLOSE_FILEprocedure, 22
- GRIBAPI_COUNT_IN_FILE() function, 22, 47
- GRIBAPI_GET procedure, 47
- GRIBAPI_GET_DATA procedure, 47
- GRIBAPI_GET_DATAprocedure, 22
- GRIBAPI_GET_SIZE() function, 22, 48
- GRIBAPI_GETprocedure, 22
- GRIBAPI_NEW_FROM_FILE() function, 22, 48
- GRIBAPI_OPEN_FILE() function, 22, 48
- GRIBAPI_RELEASE procedure, 48
- GRIBAPI_RELEASEprocedure, 22

- GRID_LAYOUT keyword
 - in WIDGET_BASE() function, 91
- GRID keyword
 - in INTERPOLATE() function, 56
 - in PY_PLOT procedure, 69
- GROUP_LEADER keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- GROUP keyword
 - in DIALOG_PICKFILE() function, 42
- GSL_EXP() function, 18, 48
- H5_GET_LIBVERSION() function, 21, 50
- H5A_CLOSE procedure, 48
- H5A_CLOSEprocedure, 21
- H5A_GET_NAME() function, 21, 48
- H5A_GET_NUM_ATTRS() function, 21, 48
- H5A_GET_SPACE() function, 21, 48
- H5A_GET_TYPE() function, 21, 48
- H5A_OPEN_IDX() function, 21, 48
- H5A_OPEN_NAME() function, 21, 48
- H5A_READ() function, 21, 48
- H5D_CLOSE procedure, 48
- H5D_CLOSEprocedure, 21
- H5D_GET_SPACE() function, 21, 49
- H5D_GET_TYPE() function, 21, 49
- H5D_OPEN() function, 21, 49
- H5D_READ() function, 21, 49
- H5F_CLOSE procedure, 49
- H5F_CLOSEprocedure, 21
- H5F_IS_HDF5() function, 21, 49
- H5F_OPEN() function, 21, 49
- H5G_CLOSE procedure, 49
- H5G_CLOSEprocedure, 21
- H5G_OPEN() function, 21, 49
- H5S_CLOSE procedure, 49
- H5S_CLOSEprocedure, 21
- H5S_GET_SIMPLE_EXTENT_DIMS() function, 21, 49
- H5T_CLOSE procedure, 49
- H5T_CLOSEprocedure, 21
- H5T_GET_SIZE() function, 21, 49
- HAS_PALETTE keyword
 - in MAGICK_PING() function, 60
- HAT keyword
 - in PLOTERR procedure, 67
- HDF_CLOSE procedure, 50
- HDF_CLOSEprocedure, 21
- HDF_OPEN() function, 21, 50
- HDF_SD_ADDDATA procedure, 50
- HDF_SD_ADDDATAprocedure, 21
- HDF_SD_ATTRFIND() function, 21, 50
- HDF_SD_ATTRINFO procedure, 50
- HDF_SD_ATTRINFOprocedure, 21
- HDF_SD_CREATE() function, 21, 50
- HDF_SD_DIMGET procedure, 50
- HDF_SD_DIMGETID() function, 21, 50
- HDF_SD_DIMGETprocedure, 21
- HDF_SD_END procedure, 50
- HDF_SD_ENDACCESS procedure, 50
- HDF_SD_ENDACCESSprocedure, 21
- HDF_SD_ENDprocedure, 21
- HDF_SD_FILEINFO procedure, 50
- HDF_SD_FILEINFOprocedure, 21
- HDF_SD_GETDATA procedure, 51
- HDF_SD_GETDATAprocedure, 21
- HDF_SD_GETINFO procedure, 51
- HDF_SD_GETINFOprocedure, 21
- HDF_SD_NAMETOINDEX() function, 21, 51
- HDF_SD_SELECT() function, 21, 51
- HDF_SD_START() function, 21, 51
- HDF_TYPE keyword
 - in HDF_SD_ATTRINFO procedure, 50
 - in HDF_SD_CREATE() function, 50
 - in HDF_SD_GETINFO procedure, 51
- HDF_VD_ATTACH() function, 21, 51
- HDF_VD_DETACH procedure, 51
- HDF_VD_DETACHprocedure, 21
- HDF_VD_FIND() function, 21, 51
- HDF_VD_GET procedure, 51
- HDF_VD_GETprocedure, 21
- HDF_VD_READ() function, 21, 51
- HDF_VG_ATTACH() function, 21, 51
- HDF_VG_DETACH procedure, 51
- HDF_VG_DETACHprocedure, 21
- HDF_VG_GETID() function, 21, 51
- HDF_VG_GETINFO procedure, 51
- HDF_VG_GETINFOprocedure, 21
- HDF_VG_GETTRS procedure, 52
- HDF_VG_GETTRSpcedure, 21
- HEADER_DEFINE keyword
 - in WRITE_BMP procedure, 92
- HEADER keyword
 - in READ_ASCII() function, 71
- HEAP_GC procedure, 52
- HEAP_GCprocedure, 15
- HELP procedure, 52
- HELP keyword
 - in APPLEMAN procedure, 33
 - in BESEL() function, 35
 - in BESELJ() function, 35
 - in BESELK() function, 35
 - in BESELY() function, 35
 - in CONGRID() function, 39
 - in DERIV() function, 41
 - in DIALOG_MESSAGE() function, 42
 - in DIALOG_PICKFILE() function, 42
 - in ESCAPE_SPECIAL_CHAR() function, 43
 - in FILE_BASENAME() function, 44
 - in FILE_COPY procedure, 44
 - in FILE_DELETE procedure, 44
 - in FILE_DIRNAME() function, 45
 - in FILE_WHICH() function, 45
 - in FINDFILE() function, 45
 - in IDL_VALIDNAME() function, 53
 - in IMAGE_STATISTICS procedure, 53
 - in PLOTERR procedure, 67
 - in PREWITT() function, 68
 - in READ_ASCII() function, 71
 - in READ_GIF procedure, 71
 - in READ_JPEG procedure, 71

- in READ_PNG() function, 71
- in ROBERTS() function, 72
- in SKIP_LUN procedure, 84
- in SMOOTH() function, 84
- in SOBEL() function, 84
- in SPL_INIT() function, 84
- in SPL_INTERP() function, 84
- in STR_SEP() function, 87
- in STRSPLIT() function, 86
- in WIDGET_BUTTON() function, 91
- in WRITE_BMP procedure, 92
- in WRITE_GIF procedure, 92
- in WRITE_JPEG procedure, 92
- in WRITE_PICT procedure, 92
- in WRITE_PNG procedure, 92
- HELPFORM() function, 52
- HELPprocedure, 15, 73
- HIGHWATER keyword
 - in MEMORY() function, 62
- HIRES keyword
 - in MAP_CONTINENTS procedure, 61
- HIST_2D() function, 19, 52, 53
- HIST_ND() function, 19, 52
- HISTOGRAM() function, 19, 52, 53
- HLS keyword
 - in TVLCT procedure, 89
- HSV keyword
 - in TVLCT procedure, 89
- HTONL keyword
 - in BYTEORDER procedure, 36
- HTONS keyword
 - in BYTEORDER procedure, 36
- HYBRID keyword
 - in NEWTON() function, 64
- hyperbolic functions, 19
- I_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- IDENTITY() function, 15, 19, 53
- IDL_BASE64() function, 26, 53
- IDL_CONSTANT() function, 20
- IDL_VALIDANEM() function, 26
- IDL_VALIDNAME() function, 12, 53
- IF, 12
- IGAMMA() function, 19, 53
- IGNORE_ACCELERATORS keyword
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- IHDR keyword
 - in WRITE_BMP procedure, 92
- IMAGE_INDEX keyword
 - in MAGICK_PING() function, 60
 - in QUERY_TIFF() function, 70
 - in READ_DICOM() function, 71
 - in READ_TIFF() function, 71
- IMAGE_STATISTICS procedure, 53
- IMAGINARY() function, 12, 53
- IMPULSENNOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- IMSL_BINOMIALCOEF() function, 19, 53
- IMSL_CONSTANT() function, 54
- IMSL_ERF() function, 18, 55
- IMSL_ZEROPOLY() function, 19, 20, 55
- IMSL_ZEROSYS() function, 56
- IN_GLOBAL keyword
 - in NCDF_ATTCOPY() function, 62
- INCHES keyword
 - in DEVICE procedure, 41
- INCLUDE_CURRENT_DIR keyword
 - in FILE_WHICH() function, 45
- INDEX keyword
 - in MAKE_ARRAY() function, 61
- INDGEN() function, 13, 15, 56
- INFINITY keyword
 - in FINITE() function, 46
- INFO keyword
 - in HELP procedure, 52
 - in MAGICK_PING() function, 60
- INFORMATION keyword
 - in DIALOG_MESSAGE() function, 42
- INFORMATIONAL keyword
 - in MESSAGE procedure, 62
- INPUT keyword
 - in HISTOGRAM() function, 52
- INSTALL_NUM keyword
 - in LMGR() function, 57
- INT keyword
 - in HDF_SD_CREATE() function, 50
- INTARR() function, 13, 15, 56
- INTEGER keyword
 - in MAKE_ARRAY() function, 61
 - in PRODUCT() function, 68
 - in TOTAL() function, 89
- INTERCHANGES keyword
 - in LUDC procedure, 59
- INTERLEAVE keyword
 - in READ_TIFF() function, 71
- INTERP keyword
 - in CONGRID() function, 39
- INTERPOL() function, 19, 56
- INTERPOLATE() function, 19, 56
- INTERPOLATE keyword
 - in SET_PLOT procedure, 75
- INVERSE keyword
 - in FFT() function, 44
 - in IMSL_ERF() function, 55
 - in WTN() function, 93
- INVERT() function, 18, 56
- IOERROR keyword
 - in MESSAGE procedure, 62
- ISHFT() function, 19, 56
- ISOTROPIC keyword
 - in CONTOUR procedure, 39
- ISSUE_ACCESS_ERROR keyword
 - in FILE_SEARCH() function, 45
- ITER keyword
 - in BESEL() function, 35
 - in BESELJ() function, 35
 - in BESELK() function, 35
 - in BESELY() function, 35
 - in RK4() function, 72
 - in VOIGT() function, 90

- ITMAX keyword
 - in BROYDEN() function, 35
 - in IMSL_ZEROSYS() function, 56
 - in NEWTON() function, 64
 - in SVDC procedure, 88
- JACOBIAN keyword
 - in IMSL_ZEROSYS() function, 56
- JENKINS_TRAUB keyword
 - in IMSL_ZEROPOLY() function, 55
- joint density function, 52
- JOURNAL procedure, 56
- JOURNALprocedure, 17
- JULIAN keyword
 - in SYSTIME() function, 88
- KBRD_FOCUS_EVENTS keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- KEYWORD_SET() function, 14, 56
- KILL_NOTIFY keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- KURTOSIS() function, 19, 56
- L64 keyword, 18
- L64_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- L64 keyword
 - in CEIL() function, 39
 - in FLOOR() function, 46
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
 - in MEMORY() function, 62
 - in ROUND() function, 72
 - in SIZE() function, 84
 - in SORT() function, 84
 - in VALUE_LOCATE() function, 90
- L64INDEGEN() function, 15
- L64INDGEN() function, 13, 56
- L64SWAP keyword
 - in BYTEORDER procedure, 36
- L_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- LA_TRIRED procedure, 56
- LA_TRIREDprocedure, 19
- LABEL keyword
 - in HDF_SD_GETINFO procedure, 51
- LAGUERRE() function, 19, 56
- LANDSCAPE keyword
 - in DEVICE procedure, 41
- LAPLACIANNOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- LAST_ITEM() function, 56
- LEGENDRE() function, 19, 57
- LENGTH_OF_HAT keyword
 - in PLOTERR procedure, 67
- LENGTH keyword
 - in N_TAGS() function, 64
 - in NCDF_ATTPUT procedure, 62
 - in STREGEX() function, 85
 - in STRSPLIT() function, 86
 - in STRTOK() function, 86
- LEVEL keyword
 - in ROUTINE_NAMES() function, 73
 - in SCOPE_VARFETCH() function, 75
- LEVELS keyword
 - in CONTOUR procedure, 39
- LIB keyword
 - in HELP procedure, 52
- LINDEGEN() function, 15
- LINDGEN() function, 13, 57
- LINE_FILL keyword
 - in POLYFILL procedure, 67
- LINEAR keyword
 - in RADON() function, 70
- LINEINTERLACE keyword
 - in MAGICK_INTERLACE procedure, 60
- LINES keyword
 - in SKIP_LUN procedure, 84
- LINESTYLE keyword
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SURFACE procedure, 87
- LINKIMAGE procedure, 57
- LINKIMAGE() function, 31
- LINKIMAGEprocedure, 38
- LIST_OF_SPECIAL_CHAR keyword
 - in ESCAPE_SPECIAL_CHAR() function, 43
- LL_ARC_DISTANCE() function, 19, 23, 57
- LMGR() function, 57
- LMHOSTID keyword
 - in LMGR() function, 57
- LNGAMMA() function, 19, 57
- LOADCT procedure, 57
- LOADCT_INTERNALGDL procedure, 58
- LOADCTprocedure, 23
- LOCALE_GET() function, 25, 59
- LOCATIONS keyword
 - in HISTOGRAM() function, 52
- LOGICAL_AND() function, 12, 59
- LOGICAL_OR() function, 12, 59
- LOGICAL_TRUE() function, 12, 59
- LON64ARR() function, 15, 59
- LONARR() function, 13, 15, 59
- LONG() function, 12, 13, 59
- LONG64() function, 12, 13, 59
- LONG64ARR() function, 13
- LONG keyword
 - in HDF_SD_CREATE() function, 50
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
 - in NCDF_ATTPUT procedure, 62
 - in NCDF_VARDEF() function, 63
 - in RANDOMN() function, 70
 - in RANDOMU() function, 70
- LSQUADRATIC keyword

in INTERPOL() function, 56
 LSWAP keyword
 in BYTEORDER procedure, 36
 LUDC procedure, 59
 LUDCprocedure, 19
 LUSOL() function, 59
 LUT keyword
 in IMAGE_STATISTICS procedure, 53

MACHAR() function, 15, 59
 MAGICK_ADDNOISE procedure, 59
 MAGICK_CLOSE procedure, 59
 MAGICK_COLORMAPSIZE() function, 60
 MAGICK_COLUMNS() function, 60
 MAGICK_CREATE() function, 60
 MAGICK_DISPLAY procedure, 60
 MAGICK_EXISTS() function, 60
 MAGICK_FLIP procedure, 60
 MAGICK_INDEXEDCOLOR() function, 60
 MAGICK_INTERLACE procedure, 60
 MAGICK_MAGICK() function, 60
 MAGICK_MATTE procedure, 60
 MAGICK_OPEN() function, 60
 MAGICK_PING() function, 60
 MAGICK_QUALITY procedure, 60
 MAGICK_QUANTIZE procedure, 61
 MAGICK_READ() function, 61
 MAGICK_READCOLORMAPRGB procedure, 61
 MAGICK_READINDEXES() function, 61
 MAGICK_ROWS() function, 61
 MAGICK_WRITE procedure, 61
 MAGICK_WRITECOLORTABLE procedure, 61
 MAGICK_WRITEFILE procedure, 61
 MAGICK_WRITEINDEXES procedure, 61
 magnitude of a complex number, 18
 MAKE_ARRAY() function, 15, 61
 MANAGED keyword
 in WIDGET_CONTROL procedure, 91
 in WIDGET_INFO() function, 91
 Mandelbrot set, 33
 MAP_CLIP_SET procedure, 61

MAP_CLIP_SETprocedure, 23
 MAP_CONTINENTS procedure, 61
 MAP_CONTINENTSprocedure, 23
 MAP_PROJ_FORWARDprocedure, 23
 MAP_PROJ_INVERSEprocedure, 23
 MAP keyword
 in MAGICK_READ() function, 61
 in TRIGRID() function, 89
 in WIDGET_BASE() function, 91
 in WIDGET_CONTROL procedure, 91
 MARK_DIRECTORY keyword
 in FILE_DIRNAME() function, 45
 in FILE_SEARCH() function, 45
 MASK keyword
 in CHECK_MATH() function, 39
 in IMAGE_STATISTICS procedure, 53
 MATCH_ALL_INITIAL_DOT keyword
 in FILE_SEARCH() function, 45
 MATCH_INITIAL_DOT keyword
 in FILE_SEARCH() function, 45
 MATRIX_MULTIPLY() function, 18, 61
 MAX() function, 18, 19, 62
 MAX1 keyword
 in HIST_2D() function, 52
 MAX2 keyword
 in HIST_2D() function, 52
 MAX_VALUE keyword
 in CONTOUR procedure, 39
 in OPLOT procedure, 66
 in PLOT procedure, 67
 in SURFACE procedure, 87
 in TRIGRID() function, 89
 MAX keyword
 in BYTSCL() function, 36
 in HIST_ND() function, 52
 in HISTOGRAM() function, 52
 in MIN() function, 62
 MAXIMUM keyword
 in IMAGE_STATISTICS procedure, 53
 MAXMOMENT keyword
 in MOMENT() function, 62

MBAR keyword
 in WIDGET_BASE() function, 91
 MDEV keyword
 in MOMENT() function, 62
 MEAN() function, 18, 19, 62
 MEAN keyword
 in IMAGE_STATISTICS procedure, 53
 MEANABSDEV() function, 19, 62
 MEDIAN() function, 19, 28, 62
 MEMORY() function, 62
 MEMORY keyword
 in HELP procedure, 52
 MEMORYprocedure, 17
 MENU keyword
 in WIDGET_BUTTON() function, 91
 MESSAGE procedure, 62
 MESSAGEprocedure, 15, 17
 MIN() function, 18, 19, 62
 MIN1 keyword
 in HIST_2D() function, 52
 MIN2 keyword
 in HIST_2D() function, 52
 MIN_VALUE keyword
 in CONTOUR procedure, 39
 in OPLOT procedure, 66
 in PLOT procedure, 67
 in SURFACE procedure, 87
 MIN keyword
 in BYTSCL() function, 36
 in HIST_ND() function, 52
 in HISTOGRAM() function, 52
 in MAX() function, 62
 MINIMUM keyword
 in IMAGE_STATISTICS procedure, 53
 MINUS_ONE keyword
 in CONGRID() function, 39
 MISSING_VALUE keyword
 in READ_ASCII() function, 71
 MISSING keyword
 in BILINEAR() function, 35
 in CONGRID() function, 39

- in INTERPOLATE() function, 56
 - in POLY_2D() function, 68
 - in TRIGRID() function, 89
- MODAL keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_INFO() function, 91
- MOMENT() function, 19, 62
- MONTH keyword
 - in PRINT procedure, 68
 - in PRINTF procedure, 68
 - in READ procedure, 70
 - in READF procedure, 70
 - in READS procedure, 70
 - in STOP procedure, 85
 - in STRING() function, 85
- MORE keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
- MTIMES keyword
 - in SAVE procedure, 75
- MULTIPLE_FILES keyword
 - in DIALOG_PICKFILE() function, 42
- MULTIPLE keyword
 - in WRITE_GIF procedure, 92
- MULTIPLICATIVEGAUSSIANNNOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- multithreading
 - in ABS() function, 33
 - in ACOS() function, 33
 - in ASIN() function, 34
 - in ATAN() function, 34
 - in CEIL() function, 39
 - in COMPLEX() function, 39
 - in CONJ() function, 39
 - in COS() function, 40
 - in COSH() function, 41
 - in EXP() function, 43
 - in FFT() function, 44
 - in FLOOR() function, 46
 - in IMAGINARY() function, 53
 - in LOGICAL_AND() function, 59
 - in LOGICAL_OR() function, 59
 - in LOGICAL_TRUE() function, 59
 - in MAGICK_WRITEINDEXES procedure, 61
 - in PRODUCT() function, 68
 - in PTRARR() function, 68
 - in ROUND() function, 72
 - in SIN() function, 83
 - in SINH() function, 83
 - in SQRT() function, 85
 - in STRCOMPRESS() function, 85
 - in STRLEN() function, 86
 - in STRLOWCASE() function, 86
 - in STRMID() function, 86
 - in STRPOS() function, 86
 - in STRPUT procedure, 86
 - in STRTRIM() function, 87
 - in STRUPCASE() function, 87
 - in TAN() function, 88
 - in TANH() function, 88
 - in TOTAL() function, 89
 - in WHERE() function, 91
- MUST_EXIST keyword
 - in DIALOG_PICKFILE() function, 42
- N_DIMENSIONS keyword
 - in SIZE() function, 84
- N_ELEMENTS() function, 14, 15, 64
- N_ELEMENTS keyword
 - in SIZE() function, 84
- N_PARAMS() function, 14, 64
- N_TAGS() function, 15, 64
- NAME keyword
 - in CREATE_STRUCT() function, 41
 - in HDF_SD_ATTRINFO procedure, 50
 - in HDF_SD_DIMGET procedure, 50
 - in HDF_SD_GETINFO procedure, 51
 - in HDF_VD_GET procedure, 51
 - in HDF_VG_GETINFO procedure, 51
- NAMED_PIPE keyword
 - in FILE_TEST() function, 45
- NAMES keyword
 - in SAVE procedure, 75
- NAN keyword
 - in BYTSCL() function, 36
 - in FINITE() function, 46
 - in HISTOGRAM() function, 52
 - in KURTOSIS() function, 56
 - in MAX() function, 62
 - in MEAN() function, 62
 - in MEANABSDEV() function, 62
 - in MIN() function, 62
 - in MOMENT() function, 62
 - in PRODUCT() function, 68
 - in SKEWNESS() function, 84
 - in SMOOTH() function, 84
 - in STDDEV() function, 85
 - in TOTAL() function, 89
 - in TVSCL procedure, 89
 - in VARIANCE() function, 90
- NATTR keyword
 - in HDF_SD_DIMGET procedure, 50
- NATTS keyword
 - in HDF_SD_GETINFO procedure, 51
- NBINS keyword
 - in HIST_ND() function, 52
 - in HISTOGRAM() function, 52
- NCDF_ATTCOPY() function, 21, 62
- NCDF_ATTDEL procedure, 62
- NCDF_ATTDELprocedure, 21
- NCDF_ATTGET procedure, 62
- NCDF_ATTGETprocedure, 21
- NCDF_ATTINQ() function, 21, 62
- NCDF_ATTNAME() function, 21, 62
- NCDF_ATTPUT procedure, 62
- NCDF_ATTPUTprocedure, 21
- NCDF_ATTRENAME procedure, 63
- NCDF_ATTRENAMEprocedure, 21
- NCDF_CLOSE procedure, 63
- NCDF_CLOSEprocedure, 21
- NCDF_CONTROL procedure, 63

- NCDF_CONTROL procedure, 21, 64
- NCDF_CREATE() function, 21, 63
- NCDF_DIMDEF() function, 21, 63
- NCDF_DIMID() function, 21, 63
- NCDF_DIMINQ procedure, 63
- NCDF_DIMINQ procedure, 21
- NCDF_DIMRENAME procedure, 63
- NCDF_DIMRENAME procedure, 21
- NCDF_EXISTS() function, 21, 63
- NCDF_INQUIRE() function, 21, 63
- NCDF_OPEN() function, 21, 63
- NCDF_VARDEF() function, 21, 63
- NCDF_VARGET procedure, 63
- NCDF_VARGET1 procedure, 64
- NCDF_VARGET1 procedure, 21
- NCDF_VARGET procedure, 21
- NCDF_VARID() function, 21, 64
- NCDF_VARINQ() function, 21, 64
- NCDF_VARPUT procedure, 64
- NCDF_VARPUT procedure, 21
- NCDF_VARRENAME procedure, 64
- NCDF_VARRENAME procedure, 21
- NCOLORS keyword
 - in LOADCT procedure, 57
- NCOMPLEMENT keyword
 - in WHERE() function, 91
- NDIMS keyword
 - in HDF_SD_GETINFO procedure, 51
- NENTRIES keyword
 - in HDF_VG_GETINFO procedure, 51
- NEWTON() function, 20, 64
- NLEVELS keyword
 - in CONTOUR procedure, 39
- NO_CHECK keyword
 - in DERIV() function, 41
- NO_CONFIRM keyword
 - in EXIT procedure, 43
- NO_COPY keyword
 - in PTR_NEW() function, 69
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_CONTROL procedure, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- NO_INTERLACE keyword
 - in HDF_VD_READ() function, 51
- NO_NEWLINE keyword
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- NO_RELEASE keyword
 - in WIDGET_BUTTON() function, 91
- NO_TYPECONV keyword
 - in ARRAY_EQUAL() function, 34
- NOAUTOMODE keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
- NOCATCH keyword
 - in SAVE procedure, 75
- NOCLEAR keyword
 - in CHECK_MATH() function, 39
- NOCLIP keyword
 - in CONTOUR procedure, 39
 - in OPLLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SURFACE procedure, 87
 - in XYOUTS procedure, 93
- NOCLOBBER keyword
 - in NCDF_CREATE() function, 63
- NODATA keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- NODISPLAY keyword
 - in APPLEMAN procedure, 33
- NOERASE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- in PLOT procedure, 67
- in SURFACE procedure, 87
- NOEXPAND_PATH keyword
 - in FILE_COPY procedure, 44
 - in FILE_DELETE procedure, 44
 - in FILE_INFO() function, 45
 - in FILE_LINES() function, 45
 - in FILE_MKDIR procedure, 45
 - in FILE_SAME() function, 45
 - in FILE_TEST() function, 45
- NOFILL keyword
 - in NCDF_CONTROL procedure, 63
- NOINTERLACE keyword
 - in MAGICK_INTERLACE procedure, 60
- NOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- NONAME keyword
 - in MESSAGE procedure, 62
- NONEXCLUSIVE keyword
 - in WIDGET_BASE() function, 91
- NOPREFIX keyword
 - in MESSAGE procedure, 62
- NOPRINT keyword
 - in MESSAGE procedure, 62
- NORM() function, 18, 64
- NORMAL keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in CONVERT_COORD() function, 40
 - in CURSOR procedure, 41
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in RANDOMN() function, 70
 - in RANDOMU() function, 70
 - in SURFACE procedure, 87
 - in TV procedure, 89
 - in XYOUTS procedure, 93
- NOSHELL keyword
 - in SPAWN procedure, 84

- NOSORT keyword
 - in FILE_SEARCH() function, 45
- NOTIFY_REALIZE keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- NOVERBOSE keyword
 - in NCDF_CONTROL procedure, 63
- NOWAIT keyword
 - in CURSOR procedure, 41
- NOWRITE keyword
 - in NCDF_OPEN() function, 63
- NOZERO keyword
 - in BYTARR() function, 35
 - in COMPLEXARR() function, 39
 - in DBLARR() function, 41
 - in DCOMPLEXARR() function, 41
 - in FLTARR() function, 46
 - in INTARR() function, 56
 - in LON64ARR() function, 59
 - in LONARR() function, 59
 - in MAKE_ARRAY() function, 61
 - in OBJARR() function, 64
 - in PTRARR() function, 68
 - in STRARR() function, 85
 - in STRUCT_ASSIGN procedure, 87
 - in UINTARR() function, 90
 - in ULON64ARR() function, 90
 - in ULONARR() function, 90
- NRECORDS keyword
 - in HDF_VD_READ() function, 51
- NRHO keyword
 - in RADON() function, 70
- NSUM keyword
 - in OPLOT procedure, 66
- NTHETA keyword
 - in RADON() function, 70
- NTOHL keyword
 - in BYTEORDER procedure, 36
- NTOHS keyword
 - in BYTEORDER procedure, 36
- NUM_ALLOC keyword
 - in MEMORY() function, 62
- NUM_DD keyword
 - in HDF_OPEN() function, 50
- NUM_FREE keyword
 - in MEMORY() function, 62
- NUM_IMAGES keyword
 - in MAGICK_PING() function, 60
- NUM_RECORDS keyword
 - in READ_ASCII() function, 71
- NX keyword
 - in RADON() function, 70
 - in TRIGRID() function, 89
- NY keyword
 - in RADON() function, 70
 - in TRIGRID() function, 89
- OBJ_CLASS() function, 15, 64
- OBJ_DESTROY procedure, 65
- OBJ_DESTROY() function, 13
- OBJ_DESTROYprocedure, 15
- OBJ_ISA() function, 15, 65
- OBJ_NEW() function, 15, 65
- OBJ_VALID() function, 15, 66
- OBJ keyword
 - in HEAP_GC procedure, 52
 - in MAKE_ARRAY() function, 61
- OBJARR() function, 13, 15, 64
- OF
 - in CASE statement, 13
 - in SWITCH statement, 13
- OFFSET keyword
 - in NCDF_VARGET procedure, 63
 - in NCDF_VARGET1 procedure, 64
 - in NCDF_VARPUT procedure, 64
- OLDFILL keyword
 - in NCDF_CONTROL procedure, 63
- OMAX keyword
 - in HISTOGRAM() function, 52
- OMIN keyword
 - in HISTOGRAM() function, 52
- ON_ERROR procedure, 66
- ON_ERRORprocedure, 15
- ON_IOERRORprocedure, 15
- OPENR procedure, 66
- OPENRprocedure, 21
- OPENU procedure, 66
- OPENUprocedure, 21
- OPENW procedure, 66
- OPENWprocedure, 21
- OPLOT procedure, 66
- OPLOTprocedure, 23
- ORDER keyword
 - in READ_JPEG procedure, 71
 - in READ_PNG() function, 71
 - in TV procedure, 89
 - in TVRD() function, 89
 - in WRITE_JPEG procedure, 92
 - in WRITE_PNG procedure, 92
- ORIENTATION keyword
 - in POLYFILL procedure, 67
 - in READ_TIFF() function, 71
 - in XYOUTS procedure, 93
- OUT_GLOBAL keyword
 - in NCDF_ATTCOPY() function, 62
- OUTPUT keyword
 - in HELP procedure, 52
- OVERPLOT keyword
 - in CONTOUR procedure, 39
- OVERWRITE_PROMPT keyword
 - in DIALOG_PICKFILE() function, 42
- OVERWRITE keyword
 - in FFT() function, 44
 - in FILE_COPY procedure, 44
 - in REFORM() function, 72
 - in REVERSE() function, 72
 - in WTN() function, 93
- PACKED keyword
 - in ASSOC() function, 34

- PARAMETERS keyword
 - in ROUTINE_INFO() function, 72
- PARENT_DIRECTORY keyword
 - in PATH_SEP() function, 67
- PARSE_URL() function, 25, 26, 66
- Pascal's triangle, 53
- PASS_METHOD keyword
 - in SAVE procedure, 75
- PATH_SEP() function, 25, 44, 67
- PATH keyword
 - in DIALOG_PICKFILE() function, 42
- PHASE keyword
 - in ATAN() function, 34
- PID keyword
 - in SPAWN procedure, 84
- PIXEL_TYPE keyword
 - in MAGICK_PING() function, 60
- PIXMAP keyword
 - in WINDOW procedure, 92
- PLANARCONFIG keyword
 - in READ_TIFF() function, 71
- PLANEINTERLACE keyword
 - in MAGICK_INTERLACE procedure, 60
- PLOT procedure, 67
- PLOTERR procedure, 67
- PLOTERRprocedure, 23
- PLOTprocedure, 23
- PLOTS procedure, 67
- PLOTSprocedure, 23
- PM procedure, 67
- PMprocedure, 15, 21
- POINT_LUN procedure, 67
- POINT_LUNprocedure, 21
- POISSON keyword
 - in RANDOMN() function, 70
 - in RANDOMU() function, 70
- POISSONNOISE keyword
 - in MAGICK_ADDNOISE procedure, 59
 - in MAGICK_PING() function, 60
- POLAR keyword
 - in OPLOT procedure, 66
- POLY() function, 19, 67
- POLY_2D() function, 28, 68
- POLY_AREA() function, 19, 68
- POLYFILL procedure, 67
- POLYFILLprocedure, 23
- POPD procedure, 68
- POPDprocedure, 25
- PORTRAIT keyword
 - in DEVICE procedure, 41
- POSITION keyword
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- PRESERVE_NULL keyword
 - in STRSPLIT() function, 86
 - in STRTOK() function, 86
- PRESERVE_TYPE keyword
 - in PRODUCT() function, 68
 - in TOTAL() function, 89
- PREWITT() function, 28, 68
- PRIMES() function, 19, 68
- PRINT procedure, 68
- PRINT keyword
 - in CHECK_MATH() function, 39
 - in FIX() function, 46
 - in STRING() function, 85
- PRINTD procedure, 68
- PRINTDprocedure, 25
- PRINTF procedure, 68
- PRINTFprocedure, 21
- PRINTprocedure, 15, 21
- PRO_SET_VALUE keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_CONTROL procedure, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- PROCEDURES keyword
 - in HELP procedure, 52
- PRODUCT() function, 18, 68
- PROGRESSIVE keyword
 - in WRITE_JPEG procedure, 92
- PROMPT keyword
 - in READ procedure, 70
 - in READF procedure, 70
- PSYM keyword
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTERR procedure, 67
 - in PLOTS procedure, 67
- PTR_FREE procedure, 68
- PTR_FREE() function, 13
- PTR_FREEprocedure, 15
- PTR_NEW() function, 15, 68, 69
- PTR_VALID() function, 15, 68, 69
- PTR keyword
 - in HEAP_GC procedure, 52
 - in MAKE_ARRAY() function, 61
- PTRARR() function, 13, 15, 68
- PTRARRprocedure, 15
- PUSHBUTTON_EVENTS keyword
 - in WIDGET_BUTTON() function, 91
- PUSHD procedure, 69
- PUSHDprocedure, 25
- PY_PLOT procedure, 69
- PY_PRINT procedure, 69
- PYTHON procedure, 69
- PYTHON() function, 14, 32, 69
- PYTHONprocedure, 14, 32
- QUADRATIC keyword
 - in INTERPOL() function, 56
- QUALITY keyword
 - in WRITE_JPEG procedure, 92
- QUERY_BMP() function, 28, 69
- QUERY_DICOM() function, 28, 69
- QUERY_GIF() function, 28, 70
- QUERY_IMAGE() function, 28, 70
- QUERY_JPEG() function, 28, 70
- QUERY_PICT() function, 28, 70
- QUERY_PNG() function, 28, 70

- QUERY_PPM() function, 28, 70
- QUERY_TIFF() function, 28, 70
- QUESTION keyword
 - in DIALOG_MESSAGE() function, 42
- QUIET keyword
 - in FILE_COPY procedure, 44
 - in FILE_DELETE procedure, 44
 - in FINDFILE() function, 45
 - in SAVE procedure, 75
- QUOTE keyword
 - in FILE_SEARCH() function, 45

- RADON() function, 28, 70
- RANDOMN() function, 20, 70
- RANDOMU() function, 20, 70
- RDWR keyword
 - in HDF_OPEN() function, 50
 - in HDF_SD_START() function, 51
- READ procedure, 70
- READ_ASCII() function, 71
- READ_ASCIIprocedure, 21
- READ_BINARY() function, 21, 71
- READ_BMP() function, 28, 71
- READ_DICOM() function, 28, 71
- READ_GIF procedure, 71
- READ_JPEG procedure, 71
- READ_JPEGprocedure, 28
- READ_PICT procedure, 71
- READ_PICTprocedure, 28
- READ_PNG() function, 28, 71, 73
- READ_TIFF() function, 28, 71
- READ_TIMEOUT keyword
 - in SOCKET procedure, 84
- READ_XWD() function, 28, 71
- READ keyword
 - in DIALOG_PICKFILE() function, 42
 - in FILE_TEST() function, 45
 - in HDF_OPEN() function, 50
 - in HDF_SD_START() function, 51
 - in HDF_VD_ATTACH() function, 51
 - in HDF_VG_ATTACH() function, 51

- READF procedure, 70
- READFprocedure, 21
- READprocedure, 21
- READS procedure, 70
- READS() function, 26
- READSprocedure, 21
- READU procedure, 70
- READUprocedure, 21
- REAL_PART() function, 12, 71
- REALIZE keyword
 - in WIDGET_CONTROL procedure, 91
- REBIN() function, 15, 19, 28, 71
- RECALL_COMMANDS() function, 71
- RECALL_COMMANDS keyword
 - in HELP procedure, 52
- RECALL_COMMANDSprocedure, 17
- RECORD_START keyword
 - in READ_ASCII() function, 71
- RECURSIVE keyword
 - in FILE_COPY procedure, 44
 - in FILE_DELETE procedure, 44
- REDEF keyword
 - in NCDF_CONTROL procedure, 63
- REF keyword
 - in HDF_VD_GET procedure, 51
 - in HDF_VG_GETINFO procedure, 51
- REFORM() function, 15, 72
- REGEX keyword
 - in STRSPLIT() function, 86
 - in STRTOK() function, 86
- REGULAR keyword
 - in FILE_TEST() function, 45
- RELAXED_STRUCTURE_ASSIGNMENT keyword
 - in RESTORE procedure, 72
- REMOVE_ALL keyword
 - in STR_SEP() function, 87
 - in STRCOMPRESS() function, 85
- REPEAT, 14
- REPEAT_COUNT keyword
 - in WRITE_GIF procedure, 92
- REPLICATE() function, 13, 15, 19, 72

- REPLICATE_INPLACE procedure, 72
- REPLICATE_INPLACEprocedure, 15, 19
- REQUIRE_DIRECTORY keyword
 - in FILE_COPY procedure, 44
- RESET keyword
 - in CPU procedure, 41
 - in MAP_CLIP_SET procedure, 61
 - in MESSAGE procedure, 62
- RESOLUTION keyword
 - in GET_SCREEN_SIZE() function, 47
- RESOLVE_ROUTINE procedure, 72
- RESOLVE_ROUTINEprocedure, 17
- RESOURCE_NAME keyword
 - in DIALOG_MESSAGE() function, 42
 - in DIALOG_PICKFILE() function, 42
 - in WIDGET_BASE() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- RESTORE procedure, 72
- RESTORE keyword
 - in CPU procedure, 41
- RESTORED_OBJECTS keyword
 - in RESTORE procedure, 72
- RESTOREprocedure, 22
- RESULT keyword
 - in APPLEMAN procedure, 33
- RETAIN keyword
 - in WINDOW procedure, 92
- RETALL procedure, 72
- RETALL keyword
 - in RETALL procedure, 72
- RETALLprocedure, 17
- RETURN_TYPE keyword
 - in CALL_EXTERNAL() function, 36
- REVERSE() function, 15, 18, 72
- REVERSE_INDICES keyword
 - in HIST_ND() function, 52
 - in HISTOGRAM() function, 52
- REVERSE_OFFSET keyword
 - in STRMID() function, 86

- in STRPOS() function, 86
- REVERSE_SEARCH keyword
 - in STRPOS() function, 86
- RGB keyword
 - in MAGICK_READ() function, 61
 - in MAGICK_WRITE procedure, 61
 - in READ_BMP() function, 71
 - in WRITE_BMP procedure, 92
- RHO keyword
 - in RADON() function, 70
- RIVERS keyword
 - in MAP_CONTINENTS procedure, 61
- RK4() function, 20, 72
- RK4JMG() function, 72
- RMIN keyword
 - in RADON() function, 70
- RNAME_MBAR keyword
 - in WIDGET_BASE() function, 91
- ROBERTS() function, 28, 72
- ROOT_DIR keyword
 - in FILEPATH() function, 44
- ROTATE() function, 15, 18, 28, 72
- ROUND() function, 18, 72
- ROUTINE_INFO() function, 17, 72, 74
- ROUTINE_NAMES() function, 17, 73
- ROUTINES keyword
 - in HELP procedure, 52
- ROW keyword
 - in WIDGET_BASE() function, 91
- RSTRPOS() function, 26, 74
- RUNTIME keyword
 - in LMGR() function, 57
- S_FUNCTIONS keyword
 - in ROUTINE_NAMES() function, 73
- S_PROCEDURES keyword
 - in ROUTINE_NAMES() function, 73
- S_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- SAMPLE keyword
 - in REBIN() function, 71
- SAVE procedure, 75
- SAVE keyword
 - in AXIS procedure, 35
- SAVEprocedure, 22
- SCALE_FACTOR keyword
 - in DEVICE procedure, 41
- SCALE keyword
 - in HDF_SD_DIMGET procedure, 50
- SCOPE_VARFETCH() function, 17, 74, 75
- SCR_XSIZE keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- SCR_YSIZE keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- SCROLL keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- SDEV keyword
 - in MOMENT() function, 62
- SEARCH_PATH keyword
 - in PATH_SEP() function, 67
- SECONDS keyword
 - in SYSTIME() function, 88
- SEM_CREATE() function, 29, 75
- SEM_DELETE procedure, 75
- SEM_DELETEprocedure, 29
- SEM_LOCK() function, 29, 75
- SEM_RELEASE procedure, 75
- SEM_RELEASEprocedure, 29
- SENSITIVE keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_CONTROL procedure, 91
- in WIDGET_DROPLIST() function, 91
- in WIDGET_LABEL() function, 92
- in WIDGET_TEXT() function, 92
- SEPARATOR keyword
 - in WIDGET_BUTTON() function, 91
- SET_BUTTON keyword
 - in WIDGET_CONTROL procedure, 91
- SET_CHARACTER_SIZE keyword
 - in DEVICE procedure, 41
- SET_DROPLIST_SELECT keyword
 - in WIDGET_CONTROL procedure, 91
- SET_PLOT procedure, 75
- SET_PLOTprocedure, 23
- SET_RESOLUTION keyword
 - in DEVICE procedure, 41
- SET_UNAME keyword
 - in WIDGET_CONTROL procedure, 91
- SET_UVALUE keyword
 - in WIDGET_CONTROL procedure, 91
- SET_VALUE keyword
 - in WIDGET_CONTROL procedure, 91
- SETENV procedure, 75
- SETENVprocedure, 25
- SH_LOCATION keyword
 - in FINDFILE() function, 45
- SH keyword
 - in SPAWN procedure, 84
- SHIFT() function, 18, 75
- SHORT keyword
 - in HDF_SD_CREATE() function, 50
 - in NCDF_ATTPUT procedure, 62
 - in NCDF_VARDEF() function, 63
- SHORTFORM keyword
 - in HELPFORM() function, 52
- SHOW_LIST keyword
 - in ESCAPE_SPECIAL_CHAR() function, 43
- SHOWFONT procedure, 75
- SHOWFONTprocedure, 24
- SIGNED keyword
 - in POLY_AREA() function, 68
- SILENT keyword

- in LOADCT procedure, 57
- SIN() function, 19, 83
- SINDGEN() function, 15, 26, 83
- SINGLE keyword
 - in HELPFORM() function, 52
 - in STRJOIN() function, 85
- SINH() function, 19, 83
- SITE_NOTICE keyword
 - in LMGR() function, 57
- SIZE() function, 12, 14, 15, 37, 84
- SIZE keyword
 - in HELPFORM() function, 52
 - in MAKE_ARRAY() function, 61
- SKEWNESS() function, 19, 84
- SKIP_LUN procedure, 84
- SKIP_LUNprocedure, 21
- SMOOTH() function, 28, 84
- SOBEL() function, 28, 84
- SOCKET procedure, 84
- SOCKET keyword
 - in FILE_TEST() function, 45
- SOCKETprocedure, 25
- SORT() function, 15, 26, 84
- SPACE keyword
 - in WIDGET_BASE() function, 91
- SPACING keyword
 - in POLYFILL procedure, 67
- SPAWN procedure, 84
- SPAWN_OPTIONS keyword
 - in FINDFILE() function, 45
- SPAWNprocedure, 25
- SPHER_HARM() function, 19, 84
- SPL_INIT() function, 19, 84
- SPL_INIT_OLD() function, 84
- SPL_INTERP() function, 19, 84
- SPL_INTERP_OLD() function, 84
- SPLINE keyword
 - in INTERPOL() function, 56
- SPLIT keyword
 - in MAP_CLIP_SET procedure, 61
- SQRT() function, 12, 18, 85

- SSWAP keyword
 - in BYTEORDER procedure, 36
- START keyword
 - in HDF_SD_ADDDATA procedure, 50
 - in HDF_SD_GETDATA procedure, 51
- STATUS keyword
 - in EXIT procedure, 43
 - in SAVE procedure, 75
- STDDEV() function, 19, 85
- STDDEV keyword
 - in IMAGE_STATISTICS procedure, 53
- STDIO_NON_FINITE keyword
 - in PRINT procedure, 68
 - in PRINTF procedure, 68
 - in STOP procedure, 85
- STDIO keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
- STIRLING keyword
 - in FACTORIAL() function, 43
- STOP procedure, 85
- STOPprocedure, 17
- STORE keyword
 - in ROUTINE_NAMES() function, 73
- STR_SEP() function, 26, 87
- STRARR() function, 13, 15, 26, 85
- STRCMP() function, 26, 85
- STRCOMPRESS() function, 26, 85
- STREAM keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
- STREGEX() function, 26, 85
- STRIDE keyword
 - in HDF_SD_ADDDATA procedure, 50
 - in HDF_SD_GETDATA procedure, 51
 - in NCDF_VARGET procedure, 63
 - in NCDF_VARPUT procedure, 64
- STRING() function, 13, 26, 85

- STRING keyword
 - in HDF_SD_CREATE() function, 50
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
- STRJOIN() function, 26, 85
- STRLEN() function, 26, 86
- STRLOWCASE() function, 86
- STRLOWERCASE() function, 26
- STRMATCH() function, 86
- STRMID() function, 26, 86
- STRPOS() function, 26, 86
- STRPUT procedure, 86
- STRPUT() function, 26
- STRSPLIT() function, 26, 86
- STRTOK() function, 26, 86
- STRTRIM() function, 26, 87
- STRUCT_ALIGN_BYTES keyword
 - in CALL_EXTERNAL() function, 36
- STRUCT_ASSIGN procedure, 87
- STRUCT_ASSIGNprocedure, 15
- STRUCTURE_NAME keyword
 - in HELPFORM() function, 52
 - in TAG_NAMES() function, 88
- STRUCTURE keyword
 - in MEMORY() function, 62
 - in SIZE() function, 84
- STRUCTURES keyword
 - in HELP procedure, 52
- STRUPCASE() function, 26, 87
- STUDENT keyword
 - in LMGR() function, 57
- SUB_RECT keyword
 - in MAGICK_READ() function, 61
 - in READ_TIFF() function, 71
- SUBDIRECTORY keyword
 - in FILEPATH() function, 44
- SUBEXPR keyword
 - in STREGEX() function, 85
- SUBSCRIPT_MAX keyword
 - in MIN() function, 62
- SUBSCRIPT_MIN keyword

- in MAX() function, 62
- SUBTITLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- SUM_OF_SQUARES keyword
 - in IMAGE_STATISTICS procedure, 53
- SUPERCLASS keyword
 - in OBJ_CLASS() function, 64
- SURFACE procedure, 87
- SURFACEprocedure, 23
- SVDC procedure, 88
- SVDCprocedure, 19
- SWAP_ENDIAN() function, 19, 21, 88
- SWAP_ENDIAN_INPLACE procedure, 88
- SWAP_ENDIAN_INPLACEprocedure, 19, 21
- SWAP_ENDIAN keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
- SWAP_IF_BIG_ENDIAN keyword
 - in BYTEORDER procedure, 36
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
 - in SWAP_ENDIAN() function, 88
 - in SWAP_ENDIAN_INPLACE procedure, 88
- SWAP_IF_LITTLE_ENDIAN keyword
 - in BYTEORDER procedure, 36
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
 - in SWAP_ENDIAN() function, 88
 - in SWAP_ENDIAN_INPLACE procedure, 88
- SWITCH, 13
- SYMLINK keyword
 - in FILE_TEST() function, 45

- SYMSIZE keyword
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
- SYNC keyword
 - in NCDF_CONTROL procedure, 63
- SYSTEM keyword
 - in ROUTINE_INFO() function, 72
- SYSTIME() function, 27, 88
- T3D keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in CONVERT_COORD() function, 40
 - in OPLOT procedure, 66
 - in PLOTS procedure, 67
 - in SURFACE procedure, 87
- T_PDF() function, 19, 89
- TAB_MODE keyword
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- TAG_NAMES() function, 15, 88
- TAG keyword
 - in HDF_VD_GET procedure, 51
 - in HDF_VG_GETINFO procedure, 51
- TAGFORM keyword
 - in HELPFORM() function, 52
- TAN() function, 19, 88
- TANH() function, 19, 88
- TEMPLATE procedure, 88
- TEMPLATE_BLANK procedure, 89
- TEMPLATE keyword
 - in READ_ASCII() function, 71
 - in READ_BINARY() function, 71
- TEMPORARY() function, 12, 13, 17, 89
- TERMINAL keyword
 - in FILEPATH() function, 44
- TEST procedure, 89
- TEST keyword

- in APPLEMAN procedure, 33
- in CONGRID() function, 39
- in DERIV() function, 41
- in DIALOG_PICKFILE() function, 42
- in ESCAPE_SPECIAL_CHAR() function, 43
- in FILE_COPY procedure, 44
- in FILE_DELETE procedure, 44
- in FILE_WHICH() function, 45
- in FINDFILE() function, 45
- in IDL_VALIDNAME() function, 53
- in IMAGE_STATISTICS procedure, 53
- in PATH_SEP() function, 67
- in PLOTERR procedure, 67
- in READ_ASCII() function, 71
- in READ_GIF procedure, 71
- in READ_JPEG procedure, 71
- in READ_PNG() function, 71
- in SAVE procedure, 75
- in SKIP_LUN procedure, 84
- in SMOOTH() function, 84
- in STR_SEP() function, 87
- in STRSPLIT() function, 86
- in WRITE_BMP procedure, 92
- in WRITE_GIF procedure, 92
- in WRITE_JPEG procedure, 92
- in WRITE_PICT procedure, 92
- in WRITE_PNG procedure, 92
- THEN, 12
- THETA keyword
 - in RADON() function, 70
- THICK keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in OPLOT procedure, 66
 - in PLOT procedure, 67
 - in PLOTS procedure, 67
 - in POLYFILL procedure, 67
 - in SURFACE procedure, 87
 - in USERSYM procedure, 90
- TICKLEN keyword
 - in AXIS procedure, 35

- in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- TITLE keyword
 - in CONTOUR procedure, 39
 - in DIALOG_MESSAGE() function, 42
 - in DIALOG_PICKFILE() function, 42
 - in PLOT procedure, 67
 - in PM procedure, 67
 - in PY_PLOT procedure, 69
 - in SURFACE procedure, 87
 - in WIDGET_BASE() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WINDOW procedure, 92
- TLB_FRAME_ATTR keyword
 - in WIDGET_BASE() function, 91
- TLB_ICONIFY_EVENTS keyword
 - in WIDGET_BASE() function, 91
- TLB_KILL_REQUEST_EVENTS keyword
 - in WIDGET_BASE() function, 91
- TLB_MOVE_EVENTS keyword
 - in WIDGET_BASE() function, 91
- TLB_SIZE_EVENTS keyword
 - in WIDGET_BASE() function, 91
- TMP keyword
 - in FILEPATH() function, 44
- TNAME keyword
 - in SIZE() function, 84
- TO_DATA keyword
 - in CONVERT_COORD() function, 40
- TO_DEVICE keyword
 - in CONVERT_COORD() function, 40
- TO_NORMAL keyword
 - in CONVERT_COORD() function, 40
- TOLF keyword
 - in BROYDEN() function, 35
 - in NEWTON() function, 64
- TOLX keyword
 - in BROYDEN() function, 35
 - in NEWTON() function, 64
- TOOLBAR keyword
 - in WIDGET_BASE() function, 91
- TOOLTIP keyword
 - in WIDGET_BUTTON() function, 91
- TOP keyword
 - in BYTSC() function, 36
- TOTAL() function, 18, 89
- TPOOL_MAXELTS keyword
 - in CPU procedure, 41
- TPOOL_MINELTS keyword
 - in CPU procedure, 41
- TPOOL_NTHREADS keyword
 - in CPU procedure, 41
- TRACE() function, 18, 89
- TRACEBACK keyword
 - in MESSAGE procedure, 62
- TRACKING_EVENTS keyword
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- TRANSFER_COUNT keyword
 - in READU procedure, 70
 - in SKIP_LUN procedure, 84
 - in WRITEU procedure, 92
- TRANSFORM keyword
 - in MAP_CLIP_SET procedure, 61
- TRANSPARENT keyword
 - in READ_PNG() function, 71
 - in WRITE_GIF procedure, 92
 - in WRITE_PNG procedure, 92
- TRANSPPOSE() function, 15, 18, 89
- TRIAL keyword
 - in LMGR() function, 57
- trigonometric functions, 18
- TRIGRID() function, 19, 89
- TRIM keyword
 - in STR_SEP() function, 87
- TRUE keyword
 - in READ_JPEG procedure, 71
 - in TV procedure, 89
- in TVRD() function, 89
 - in WRITE_JPEG procedure, 92
- TRUECOLOR keyword
 - in MAGICK_QUANTIZE procedure, 61
- TT_FONT keyword
 - in SHOWFONT procedure, 75
- TV procedure, 89
- TV() function, 23
- TVLCT procedure, 89
- TVLCT() function, 23
- TVprocedure, 15, 67
- TVRD() function, 23, 67, 89
- TVSCL procedure, 89
- TVSCL() function, 23
- TWO_PASS_QUANTIZE keyword
 - in READ_JPEG procedure, 71
- TYPE keyword
 - in FIX() function, 46
 - in HDF_SD_ATTRINFO procedure, 50
 - in HDF_SD_GETINFO procedure, 51
 - in INDGEN() function, 56
 - in MAGICK_PING() function, 60
 - in MAKE_ARRAY() function, 61
 - in PLOTERR procedure, 67
 - in SIZE() function, 84
- UI_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- UINDGEN() function, 13, 15, 89
- UINT() function, 12, 13, 90
- UINT keyword
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
- UINTARR() function, 13, 15, 90
- UL64_VALUE keyword
 - in CALL_EXTERNAL() function, 36
- UL64 keyword
 - in FACTORIAL() function, 43
 - in INDGEN() function, 56
 - in MAKE_ARRAY() function, 61
- UL64INDGEN() function, 13, 15, 90

UL_VALUE keyword
 in CALL_EXTERNAL() function, 36
 ULINDGEN() function, 13, 15, 90
 ULON64ARR() function, 13, 15, 90
 ULONARR() function, 13, 15, 90
 ULONG() function, 12, 13, 90
 ULONG64() function, 12, 13, 90
 ULONG keyword
 in INDGEN() function, 56
 in MAKE_ARRAY() function, 61
 UNAME keyword
 in WIDGET_BASE() function, 91
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92
 UNIFORM keyword
 in RANDOMN() function, 70
 in RANDOMU() function, 70
 UNIFORMNOISE keyword
 in MAGICK_ADDNOISE procedure, 59
 in MAGICK_PING() function, 60
 UNIQ() function, 15, 18, 26, 90
 UNIT keyword
 in HDF_SD_GETINFO procedure, 51
 in READ_JPEG procedure, 71
 in SPAWN procedure, 84
 in WRITE_JPEG procedure, 92
 UNITS keyword
 in WIDGET_BASE() function, 91
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92
 UNLIMITED keyword
 in NCDF_DIMDEF() function, 63
 UNLOAD keyword
 in CALL_EXTERNAL() function, 36
 UNTIL, 14
 UP keyword
 in CURSOR procedure, 41

UPPER keyword
 in LA_TRIED procedure, 56
 USER_INPUT keyword
 in WRITE_GIF procedure, 92
 USERSYM procedure, 90
 USEUNIT keyword
 in SAVE procedure, 75
 UTC keyword
 in SYSTIME() function, 88
 UVALUE keyword
 in WIDGET_BASE() function, 91
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92
 VALID keyword
 in WIDGET_INFO() function, 91
 VALUE_LOCATE() function, 19, 90
 VALUE keyword
 in CALL_EXTERNAL() function, 36
 in MAKE_ARRAY() function, 61
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92
 VARIABLES keyword
 in ROUTINE_NAMES() function, 73
 VARIANCE() function, 19, 90
 VARIANCE keyword
 in IMAGE_STATISTICS procedure, 53
 VARSTATUS keyword
 in SAVE procedure, 75
 VAX_FLOAT keyword
 in OPENR procedure, 66
 in OPENU procedure, 66
 in OPENW procedure, 66
 VECTOR_ENABLE keyword
 in CPU procedure, 41
 VECTOR keyword
 in IMAGE_STATISTICS procedure, 53

VERBOSE keyword
 in DIALOG_PICKFILE() function, 42
 in ESCAPE_SPECIAL_CHAR() function, 43
 in FILE_COPY procedure, 44
 in FILE_DELETE procedure, 44
 in FINDFILE() function, 45
 in HEAP_GC procedure, 52
 in IMAGE_STATISTICS procedure, 53
 in NCDF_CONTROL procedure, 63
 in READ_ASCII() function, 71
 in READ_PNG() function, 71
 in READ_TIFF() function, 71
 in RESTORE procedure, 72
 in SAVE procedure, 75
 in SMOOTH() function, 84
 in STRUCT_ASSIGN procedure, 87
 in WRITE_PNG procedure, 92
 VERSION keyword
 in WIDGET_INFO() function, 91
 VM keyword
 in LMGR() function, 57
 VOIGT() function, 19, 90
 WAIT procedure, 90
 WAIT keyword
 in CURSOR procedure, 41
 WAITprocedure, 25
 WDELETE procedure, 91
 WDELETEprocedure, 23
 WEIGHT_SUM keyword
 in IMAGE_STATISTICS procedure, 53
 WEIGHTED keyword
 in IMAGE_STATISTICS procedure, 53
 WHERE() function, 15, 34, 91
 WHILE, 14
 WIDGET_BASE() function, 30, 91
 WIDGET_BUTTON() function, 30, 91
 WIDGET_CONTROL procedure, 91
 WIDGET_CONTROLprocedure, 30
 WIDGET_DROPLIST() function, 30, 91
 WIDGET_EVENT() function, 30, 91

- WIDGET_INFO() function, 30, 91
- WIDGET_LABEL() function, 30, 92
- WIDGET_TEXT() function, 30, 92
- WIDTH keyword
 - in HELPFORM() function, 52
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SOCKET procedure, 84
 - in XYOUTS procedure, 93
- WINDOW procedure, 92
- WINDOW_STATE keyword
 - in DEVICE procedure, 41
- WINDOWprocedure, 23
- WORDS keyword
 - in TVRD() function, 89
- WRAP keyword
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- WRITE_BMP procedure, 92
- WRITE_BMPprocedure, 28
- WRITE_GIF procedure, 92
- WRITE_JPEG procedure, 92
- WRITE_JPEGprocedure, 28
- WRITE_PICT procedure, 92
- WRITE_PICTprocedure, 28
- WRITE_PNG procedure, 92
- WRITE_PNGprocedure, 28, 73
- WRITE_TIMEOUT keyword
 - in SOCKET procedure, 84
- WRITE keyword
 - in DIALOG_PICKFILE() function, 42
 - in FILE_TEST() function, 45
 - in HDF_OPEN() function, 50
 - in HDF_VD_ATTACH() function, 51
 - in HDF_VG_ATTACH() function, 51
 - in NCDF_OPEN() function, 63
- WRITEprocedure, 21
- WRITEU procedure, 92
- WSET procedure, 92
- WSETprocedure, 23
- WSHOW procedure, 92
- WSHOWprocedure, 23
- WTN() function, 20, 93
- X_BITMAP_EXTRA keyword
 - in WIDGET_BUTTON() function, 91
- X_SCROLL_SIZE keyword
 - in WIDGET_BASE() function, 91
- XAXIS keyword
 - in AXIS procedure, 35
- XCHARSIZE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- XDR keyword
 - in OPENR procedure, 66
 - in OPENU procedure, 66
 - in OPENW procedure, 66
 - in SAVE procedure, 75
- XDRTOD keyword
 - in BYTEORDER procedure, 36
- XDRTOF keyword
 - in BYTEORDER procedure, 36
- XGRIDSTYLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
- XGUESS keyword
 - in IMSL_ZEROSYS() function, 56
- XLABEL keyword
 - in PY_PLOT procedure, 69
- XLOG keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in PLOTERR procedure, 67
 - in SURFACE procedure, 87
- XMANAGER_ACTIVE_COMMAND keyword
 - in WIDGET_CONTROL procedure, 91
- XMANAGER_BLOCK keyword
 - in WIDGET_EVENT() function, 91
 - in WIDGET_INFO() function, 91
- XMARGIN keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- XMIN keyword
 - in RADON() function, 70
- XMINOR keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- XOFFSET keyword
 - in DEVICE procedure, 41
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
- XPAD keyword
 - in WIDGET_BASE() function, 91
- XPOS keyword
 - in WINDOW procedure, 92
- XRANGE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in PLOTERR procedure, 67
 - in SURFACE procedure, 87
- XSIZE keyword
 - in APPLEMAN procedure, 33
 - in DEVICE procedure, 41
 - in TV procedure, 89
 - in WIDGET_BASE() function, 91
 - in WIDGET_BUTTON() function, 91
 - in WIDGET_DROPLIST() function, 91
 - in WIDGET_LABEL() function, 92
 - in WIDGET_TEXT() function, 92
 - in WINDOW procedure, 92

XSTYLE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTHICK keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTICK_GET keyword
 in CONTOUR procedure, 39
 in SURFACE procedure, 87

XTICKFORMAT keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTICKINTERVAL keyword
 in AXIS procedure, 35
 in SURFACE procedure, 87

XTICKLAYOUT keyword
 in SURFACE procedure, 87

XTICKLEN keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTICKNAME keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in SURFACE procedure, 87

XTICKS keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTICKUNITS keyword
 in SURFACE procedure, 87

XTICKV keyword
 in CONTOUR procedure, 39

 in SURFACE procedure, 87

XTITLE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XTYPE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

XYOUTS procedure, 93

XYOUTSprocedure, 23

Y_SCROLL_SIZE keyword
 in WIDGET_BASE() function, 91

YAXIS keyword
 in AXIS procedure, 35

YCHARSIZE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

YGRIDSTYLE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in SURFACE procedure, 87

YLABEL keyword
 in PY_PLOT procedure, 69

YLOG keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in PLOTERR procedure, 67
 in SURFACE procedure, 87

YMARGIN keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

YMIN keyword

 in RADON() function, 70

YMINOR keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in SURFACE procedure, 87

YNOZERO keyword
 in AXIS procedure, 35
 in PLOT procedure, 67

YOFFSET keyword
 in DEVICE procedure, 41
 in WIDGET_BASE() function, 91
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92

YP0 keyword
 in SPL_INIT() function, 84
 in SPL_INIT_OLD() function, 84

YPAD keyword
 in WIDGET_BASE() function, 91

YPN_1 keyword
 in SPL_INIT() function, 84
 in SPL_INIT_OLD() function, 84

YPOS keyword
 in WINDOW procedure, 92

YRANGE keyword
 in AXIS procedure, 35
 in CONTOUR procedure, 39
 in PLOT procedure, 67
 in PLOTERR procedure, 67
 in SURFACE procedure, 87

YSIZE keyword
 in APPLEMAN procedure, 33
 in DEVICE procedure, 41
 in TV procedure, 89
 in WIDGET_BASE() function, 91
 in WIDGET_BUTTON() function, 91
 in WIDGET_DROPLIST() function, 91
 in WIDGET_LABEL() function, 92
 in WIDGET_TEXT() function, 92

- in WINDOW procedure, 92
- YSTYLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTHICK keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTICK_GET keyword
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
- YTICKFORMAT keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTICKINTERVAL keyword
 - in AXIS procedure, 35
 - in SURFACE procedure, 87
- YTICKLAYOUT keyword
 - in SURFACE procedure, 87
- YTICKLEN keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTICKNAME keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
- YTICKS keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTICKUNITS keyword
 - in SURFACE procedure, 87
- YTICKV keyword
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
- YTITLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YTYPE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- YUV keyword
 - in MAGICK_QUANTIZE procedure, 61
- Z_BUFFERING keyword
 - in DEVICE procedure, 41
- Z keyword
 - in XYOUTS procedure, 93
- ZCHARSIZE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZENITY_NAME keyword
 - in DIALOG_MESSAGE() function, 42
 - in DIALOG_PICKFILE() function, 42
- ZENITY_PATH keyword
 - in DIALOG_MESSAGE() function, 42
 - in DIALOG_PICKFILE() function, 42
- ZENITY_SEP keyword
 - in DIALOG_PICKFILE() function, 42
- ZERO_LENGTH keyword
 - in FILE_TEST() function, 45
- ZGRIDSTYLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZLOG keyword
 - in CONTOUR procedure, 39
- in SURFACE procedure, 87
- ZMARGIN keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZMINOR keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZRANGE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZSTYLE keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZTHICK keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZTICK_GET keyword
 - in CONTOUR procedure, 39
 - in SURFACE procedure, 87
- ZTICKFORMAT keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39
 - in PLOT procedure, 67
 - in SURFACE procedure, 87
- ZTICKINTERVAL keyword
 - in SURFACE procedure, 87
- ZTICKLAYOUT keyword
 - in SURFACE procedure, 87
- ZTICKLEN keyword
 - in AXIS procedure, 35
 - in CONTOUR procedure, 39

in PLOT procedure, 67
in SURFACE procedure, 87
ZTICKNAME keyword
in AXIS procedure, 35
in CONTOUR procedure, 39
in SURFACE procedure, 87
ZTICKS keyword
in AXIS procedure, 35
in CONTOUR procedure, 39
in PLOT procedure, 67

in SURFACE procedure, 87
ZTICKUNITS keyword
in SURFACE procedure, 87
ZTICKV keyword
in CONTOUR procedure, 39
in SURFACE procedure, 87
ZTITLE keyword
in AXIS procedure, 35
in CONTOUR procedure, 39
in PLOT procedure, 67

in SURFACE procedure, 87
ZTYPE keyword
in CONTOUR procedure, 39
in SURFACE procedure, 87
ZVALUE keyword
in AXIS procedure, 35
in CONTOUR procedure, 39
in PLOT procedure, 67
in SURFACE procedure, 87

Bibliography

- [1] Fundation, F. S.: GNU General Public License, version 2, URL <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>, 1991.
- [2] Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., and Rossi, F.: GNU Scientific Library Reference Manual - Third Edition (v1.12), Network Theory Ltd., URL <http://www.gnu.org/software/gsl/manual/>, 2009. {7}
- [3] Markwardt, C.: Non-linear Least-squares Fitting in IDL with MPFIT, in: Astronomical Society of the Pacific Conference Series, edited by Bohlender, D., Durand, D., and Dowler, P., vol. 411 of *Astronomical Society of the Pacific Conference Series*, URL <http://cdsads.u-strasbg.fr/abs/2009ASPC..411..251M>, 2009. {19}
- [4] Paoli, S.: C++ Coding Standard Specification, Tech. rep., CERN European Laboratory for Particle Physics, URL <http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/Programming/CodingStandard/c++standard.pdf>, 2000. {95}
- [5] Snyder, J.: Map projections—A working manual, Tech. Rep. 1395, U.S. Geological Survey, URL http://pubs.er.usgs.gov/djvu/PP/pp_1395.djvu, 1987. {57}
- [6] van Rossum, G. and Fred L. Drake, J.: The Python Language Reference Manual, Network Theory Ltd., URL <http://docs.python.org/reference/>, 2006. {32}
- [7] Wessel, P. and Smith, W. H. F.: A global, self-consistent, hierarchical, high-resolution shoreline database, *J. Geophys. Res.*, 101, 8741–8743, doi:10.1029/96JB00104, 1996. {61}
- [8] Wolcott, N. and Hilsenrath, J.: Tables of coordinates for Hershey's repertory of occidental type fonts and graphic symbols. A contribution to computer typesetting techniques., NBS special publication 424, National Bureau of Standards, 1975. {24}