IDL语言基础及程序设计（1）

辜智慧

Email: gzh@ires.cn

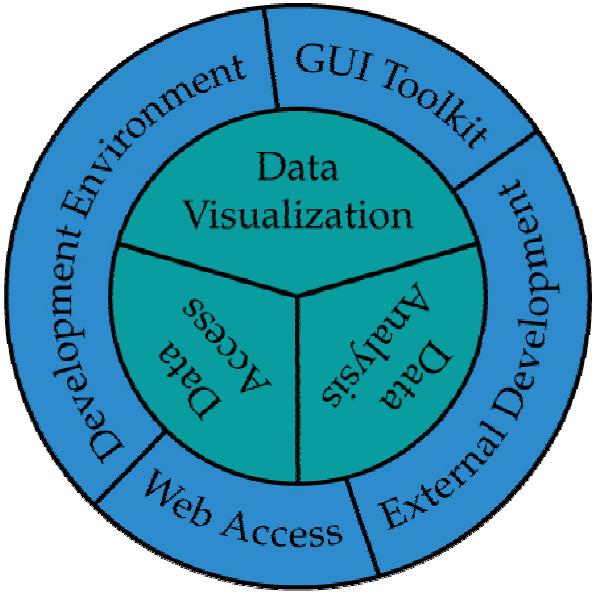
IDL参考资料

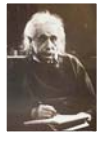
- online HELP
- http://www.rsi.com
- http://www.supresoft.com.cn
- http://www.idlworld.com
- 闫殿武，IDL可视化工具入门与提高，机械工业出版社，2003。定价：42.00元

What is IDL?

Interactive Data Language

——交互式数据语言



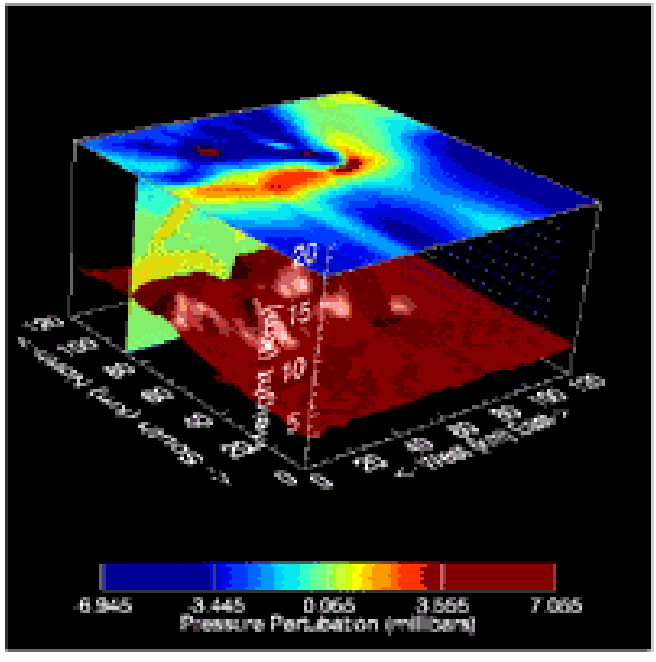


- **An interactive environment for exploring and** *analyzing data*

- **A high-level language for rapid prototyping and** *application development*

IDL 功能简介

IDL是进行数据分析、可视化及跨平台应用开发的最佳选择。**IDL**集可视、交互分析、大型商业开发为一体，为您提供了最完善、最灵活、最有效的开发环境。

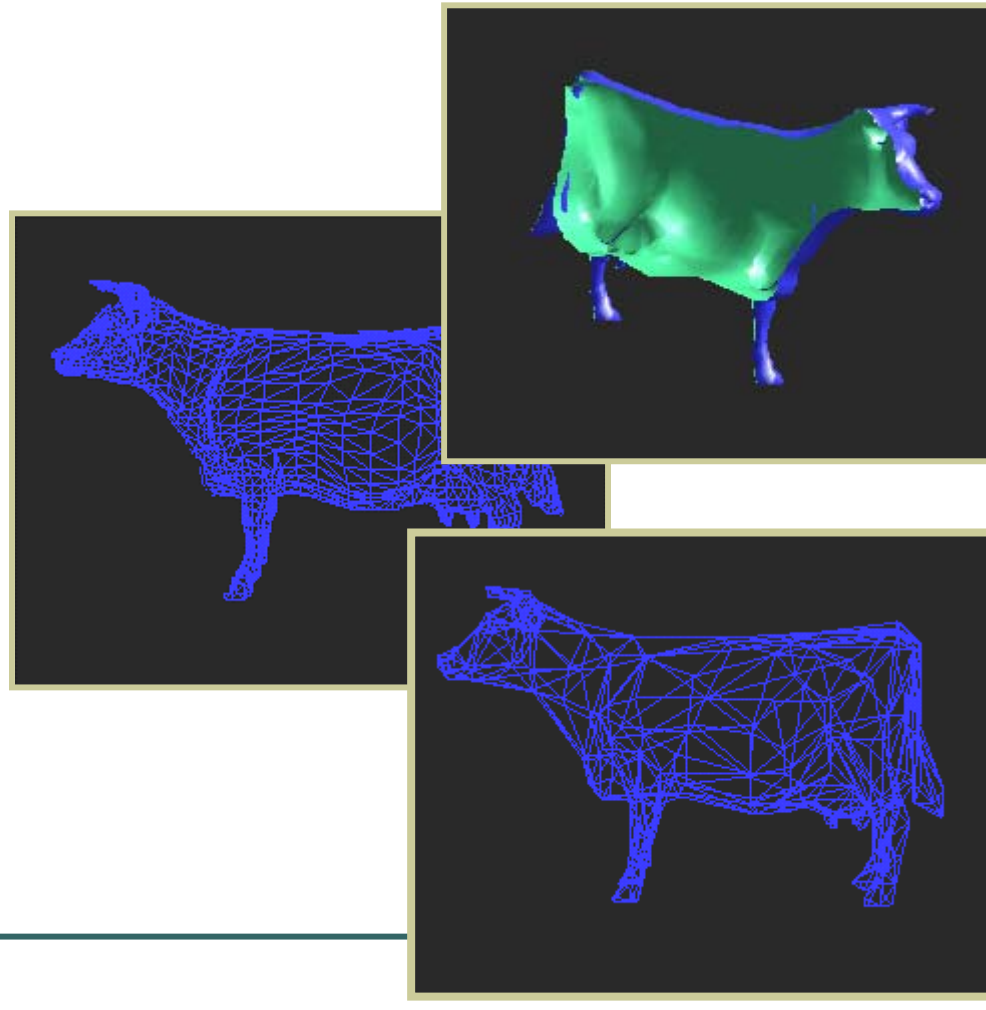


IDL 能够做到

- 快捷的交互式数据可视化
- 语法简单的**4GL**语言
- 面向矩阵的语言
  - 代码少
  - 速度快
- 迅速生成结果
  - 没有冗长的编辑-编译-链接周期

IDL采用的新技术

- 面向对象技术
- 拖放式界面控制
- 跨平台开发环境
- **COM/ActiveX**
- **OpenGL**
- **ODBC**
- **Java**



IDL的特点

- 通过灵活方便的I/O使您可以分析任何数据
- 采用OpenGL技术，可加速交互式的2D及3D数据分析、图像处理及可视化.
- 具有完善的图像处理软件包，例如感兴趣区(ROI)及一整套图像分析工具，地图投影及转换软件包，使您开发GIS易如反掌
- IDL的面向对象技术为您提供最好的解决方案.
- 完善的数学分析和统计软件包提供强大的科学计算模型.

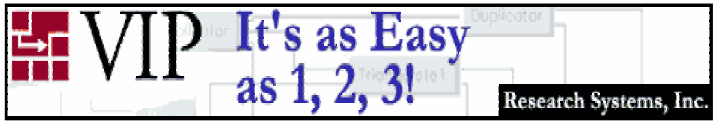
IDL的特点

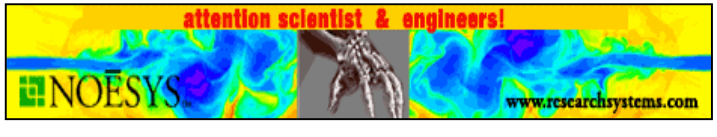
- 用 IDL DataMiner 可快速访问、查询并管理与ODBC兼容的数据库.
- IDL GUIBuilder 可以迅速开发跨平台的用户图形界面(GUI),而无需熟悉IDL控件编程知识.
- IDL的 ActiveX控件将您的IDL应用开发集成到与COM兼容的环境中。

IDL应用领域

- 地球科学
- 医学影像
- 图像处理
- 软件开发
- 大学教学
- 实验室
- 测试技术
- 天文
- 信号处理
- 防御工程
- 数学分析
- 统计

基于IDL的系列产品





IDL - Interactive Data Language

ION - IDL On the Net

VIP - Visual IDL Programming

ENVI - Environment for Visualizing images

RiverTools - Analysis for digital terrain and river network modeling

NOESYS - Organize, visualize & share HDF data

IDL语法基础

- 变量设定
  - 标量 scalar
  - 数组 array （1-8维）
  - 结构变量 structure
- IDL命令解析
  - 命令格式
  - 命令参数
  - 命令日志

变量设定

1. 变量及其属性

- 变量无须事先声明
- 变量名必须以字母开头，可包括数字、下划线、美元符号。最长可达255个字符
- 动态改变变量的属性
- 变量的大小取决于计算机配置和操作系统

变量设定

2. 基本数据类型

数据类型	字节	值域	创建	类型函数
字节	1	0 ~ 255	0B	byte()
16 位有符号整型	2	-32,768 ~ 32,767	0	fix()
32 位有符号整型	4		0L	long()
64 位有符号整型	8		0LL	long64()
16 位无符号整型	2	0 ~ 65535	0U	uint()
32 位无符号整型	4	0 ~ 2^32-1	0UL	ulong()
64 位无符号整型	8	0 ~ 2^64-1	0ULL	ulong64()
浮点型	4	±10^38	0.0	float()
双精度浮点型	8		0.0D	double()
复数	8		complex(0.0,0.0)	complex()
双精度复数	16		complex(0.0D,0.0D)	dcomplex()
字符串	0 ~ 32767		‘’ 或 “”	string()
指针	4		ptr_new()	-
对象	4		obj_new()	-

注意IDL中整型变量为其他编程语言中的短整型，“L”表示为长整型

变量设定
3. 数组

IDL是面向矩阵的语言，几乎所有运算都可以在数组上使用。

数组表达：array[n,m] 表示n列m行（与其他语言有别），按行排列, 0为下标起点
数组引用：array[subscript]，或（array）[subscript]
下标语法：e、e0:e1、e:*、*、array。
array=make_array(10,10,/integer)，sub=indgen(12)
合法的下标表示：array[5,5]、array[2:3,5]、array[*,4]、array[*,5:8]、array[4,4:*]、array[sub]、A[[1,3,5],7:9]
reform()：array[4,4:*]为1列4行（列向量），reform(array[4,4:*])则为4列1行（行向量）
常数的数组表示：var=5，则var[0]=5（合法！）
赋值：array[[2, 4, 6],5]=[4, 16, 36]
where()：array[where(array lt 0)]=-999
数学运算：与普通变量基本相同。
* 和 / ：表示两个同维数数组对应元素运算
arr1=indgen(5)+1，arr2=arr1。则：arr1*arr2=[1,4,9,16,25]，arr1/arr2=[1,1,1,1]
 # 和 ##：矩阵运算 arr1(n1,m)#arr2(m,n2)=arr(n1,n2)，
arr1(n,m1)##arr2(m2,n)=arr(m2,m1)
数组串连：arr1(5,6)，arr2(5,2)。则：arr3=[[arr1],[arr2]]为（5,8）
注意： arr3=[arr1,arr2]不合法！（一维除外）

变量设定

4. 数组常用函数

数据类型	初始化函数	产生索引值函数创建
字节	bytarr	bindgen
16 位有符号整型	intarr	indgen
32 位有符号整型	lonarr	lindgen
64 位有符号整型	lon64arr	l64indgen
16 位无符号整型	uintarr	uindgen
32 位无符号整型	ulonarr	ulindgen
64 位无符号整型	ulon64arr	ul64indgen
浮点型	fltarr	findgen
双精度浮点型	dblarr	dindgen
复数	complexarr	cindgen
双精度复数	dcomplexarr	dcindgen
字符串	strarr	slindgen
指针	ptrarr	-
对象	objarr	-

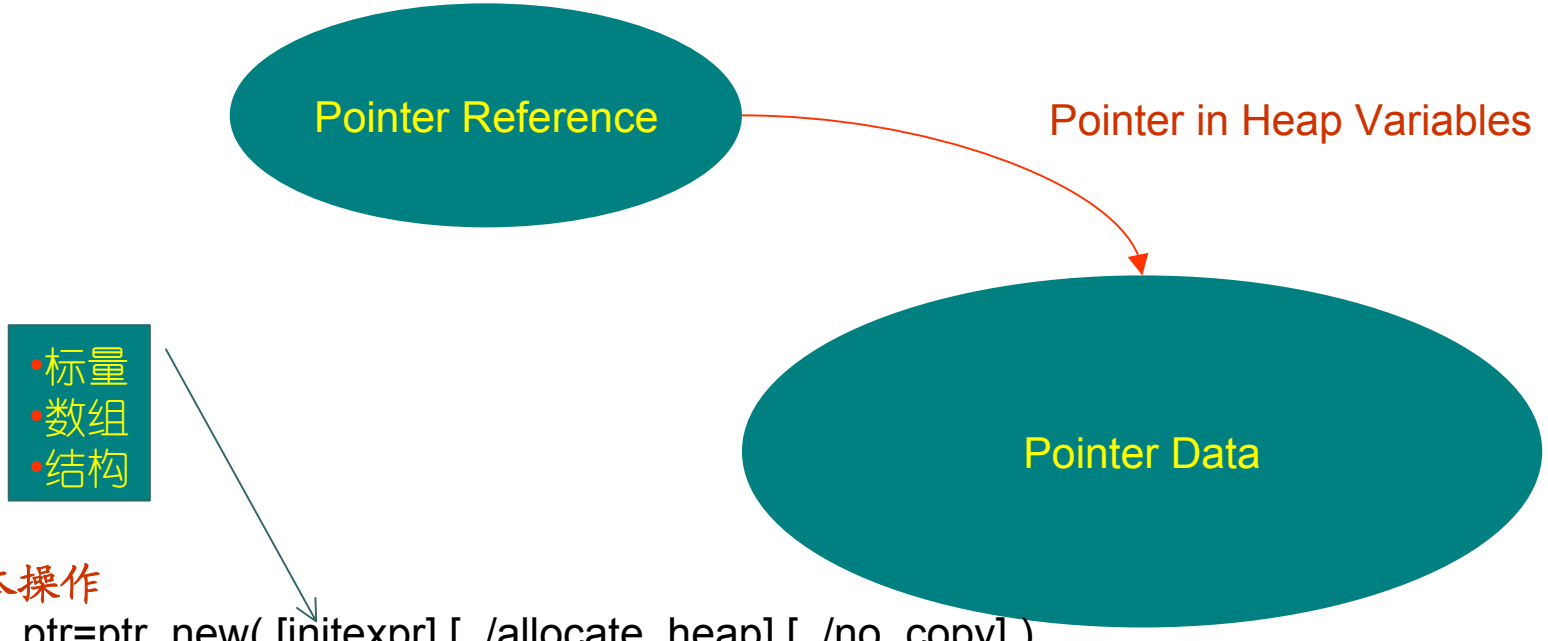
变量设定
5. 结构变量

结构变量是一种复合变量，它可以将多种类型的数据存储在一个变量中，

- 类型及定义
  命名结构：dot={PIXEL ,x:128 ,y:236 ,color:bytarr(3)}，定义后可使用FIXEL定义其他结构
  dot1={PIXEL ,x:58 ,y:46 ,color:[255,0,255]}、
  dot2={PIXEL ,58 ,46 ,[255,0,255]}、dot3= {PIXEL}
  匿名结构：person={name:'jack' ,id:123456L}，定义后无固定结构，可任意改变
  person={name:'jack' ,id:123456L ,phone:'123-4567'}
- 引用
  变量引用：使用变量名或变量在结构中的位置索引。如：dot.x或dot.(0)
  数组变量：s={arr:indgen(10)}，则s.arr=10将数组所有元素赋值为10。
- 结构数组
  定义：dotarr=replicate({PIXEL} ,10)，或dotarr=replicate(dot ,10)
  引用：dotarr[1].x=10、dotarr.x=10将所有结构的x赋值为10、dotarr.y=indgen(10)
- 结构中的变量的类型和（数组）大小
  结构定义后，各变量的数据类型以及数组变量的维数均不可改变。当使用中出现不一致时向原类型转换，不能转换时报错。
  var=dot.x*1.0=128.0，为浮点，而dot.x=dot仍为整型。
  s.arr=-indgen(8)会改变s.arr中前8个元素的值，而s.arr=-indgen(11)会出错。
- 结构继承
  dot3d={POINT ,INHERITS PIXEL ,z:0}
- 常用函数　creat_struct()、n_tags()、tag_names()、struct_assign()

变量设定
6. 指针变量

IDL 的指针与其他语言的指针有很大的不同，它不是指向存储的地址而仅仅是一个轻型的指向一个堆变量的引用（指针变量）。堆变量可以动态分配（数据类型和数组维数），这意味着传递指针变量就相当于传递动态数据。



指针基本操作

创建: ptr=ptr_new([initexpr] [, /allocate_heap] [, /no_copy])

释放: ptr_free ,ptr

变量设定
7. 指针基本命令

- 标量指针
  - 创建：v=5.5，p=ptr_new(v)
  - 引用：print , p ,*p；p1=p，*p1=20，print ,*p
- 数组指针
  - 创建：arr=findgen(10)，p=ptr_new(arr)
  - 引用：print ,(*p)[5]
- 结构指针
  - 创建：s={name:'joe' ,age:40 ,height:180} ， p=ptr_new(s)
  - 引用：print , (*p).name
- 结构内指针
  - 创建：rec={lon:120 ,lat:20 ,data:ptr_new(findgen(2,10))} ， p=ptr_new(rec)
  - 引用：* (*p).data=findgen(2,20)
- 特殊指针
  - Null指针：nptr=ptr_new()，仅定义一个指针，并不指向一个堆变量。引用时需重新定义指针。
  - Empty指针：eptr=ptr_new(/allocate_heap)，定义一个指向一个堆变量的指针，但并未定义变量，引用时可以直接定义变量
- 指针释放
  - ptr_free ,ptr
- 相关函数
  - ptr_valid()： ptr_valid(nptr)=0， ptr_valid(eptr)=1
  - heap_gc：释放没有引用的堆变量
- 指针数组
  - ptrarr(d1, ... , d8 [, /allocate_heap])

IDL命令解析

1. 命令格式

——宽松的语法检查机制

- 分隔符为“,”，而非空格
- 不分大小写
- $ 为续行符
- ；后面是注释

```
A=dist(100)          Zvalue=0.5
Tvscl,a              Contour,a,nlevels=10
Shade_surf,a         Contour,a,nlevels=10,$,/follow
```

IDL命令解析
2. 命令参数

- 位置参数：在命令名右边，有严格的顺序限制。通常用于必选参数。
  - 例如： Contour, peak, lon, lat …
  - 错误： Contour, lon, peak, lat …
- 关键字参数：关键字参数与位置无关，且可以与位置参数混合位置。通常放在位置参数之后，用于可选参数。
  - 例如： Contour, peak, levels=vals, lon, /follow, lat, ...

命令帮助：

IDL>?

IDL命令解析

3. 命令日志

将在命令行里面输入的命令保存为日志或记录，该文件以pro格式保存，可作为一个IDL批处理文件

例如，创建日志文件commands.pro

IDL>Journal, ‘commands’

关闭日志文件

IDL>Journal

IDL编程基础

- 批处理文件\主程序\过程\函数
- 参数传递
- 错误处理
- 编译与运行
- 输入输出
- 常用控制语法

IDL程序
1. 批处理

- 主体：由一系列IDL命令组成

  例如
  ```
  thisfiles=findfile(*.img,count=numfiles)
  Print,'number of files found:',numfiles
  ....
  ```
- 运行方式：以IDL>@batchfile方式运行。
- 编译：批处理文件运行时并不编译，因此使用控制结构时必须大量使用续行符（$）,给书写、理解造成困难。

IDL程序
2. 主程序

- 主体：与批处理相似，但必须以end结束，有两种表现形式

（1）
```
...
end
```

（2）
```
pro test
    程序体
...
end
```

文件名为test.pro，没有名称的主程序必须放在程序最后面。

- 运行方式：以IDL>.run profile方式运行。
- 编译：主程序运行时先编译，因此可以正常使用控制结构。源代码编译后，直接执行与文件名同名的主程序

IDL程序
3. 过程

- 主体：与主程序相似，但必须以pro proname开始，以end结束。

  pro 过程名称，变量V1,V2, ... ,k1=k1,k2=k2
  程序体
  end
  keyword起重要作用
- 运行方式：以IDL>proname方式运行
- 编译：可以先运行IDL>.compile proname，编译但不运行

IDL程序

4. 函数

- 主体：与过程相似，但以function fnname开始，以end结束，并以return语句返回一个IDL变量。

```
Function test    变量 V1,V2, …  ,
                 关键字 k1=k1,k2=k2
        程序体
return, Value
end
```

- 运行方式：以IDL>ret=fnname(para_list)方式运行。

IDL程序

5. 注意事项

- 在IDL系统中，一个过程或函数即为一个新的IDL命令。
- 变量作用范围：批处理和主程序方式的变量为全局变量，可以在IDL开发环境中使用。过程和函数的变量为局部变量，只在过程和函数运行过程中有效。

参数传递

位置参数：在参数列表中按位置列出参数名，严格的顺序限制。通常用于必选参数。

定义：pro batch ,para1 ,para2 ,...

调用：IDL->batch ,para1 ,para2 ,…

关键字参数：关键字参数与位置无关，且可以与位置参数混合位置。通常放在位置参数之后，用于可选参数。

定义：pro batch ,keywordname=keywordsymbol ,...

调用：IDL->batch ,keywordname=keywordsymbol ,…

IDL->batch ,/keywordname

注意：keywordname用于定义，keywordsymbol用于调用。

引用传递和值传递：所有变量为引用传递，其值会被修改。系统变量、下标变量、表达式和常量均为值传递，原变量的值不被修改。

参数传递了吗？传递了什么？

n_params()：返回位置参数的个数

keyword_set()：关键字参数为不为0常量或已定义的引用传递时返回1，否则返回0

arg_present()：关键字参数为引用传递时返回1（无论是否定义），否则返回0

n_elements()：关键字参数未传递或未定义返回0，否则返回非0数

错误处理

on_ioerror： 当出现I/O错误时，跳转指定的语句。两种用途：跳过错误返回或跳过错误继续。

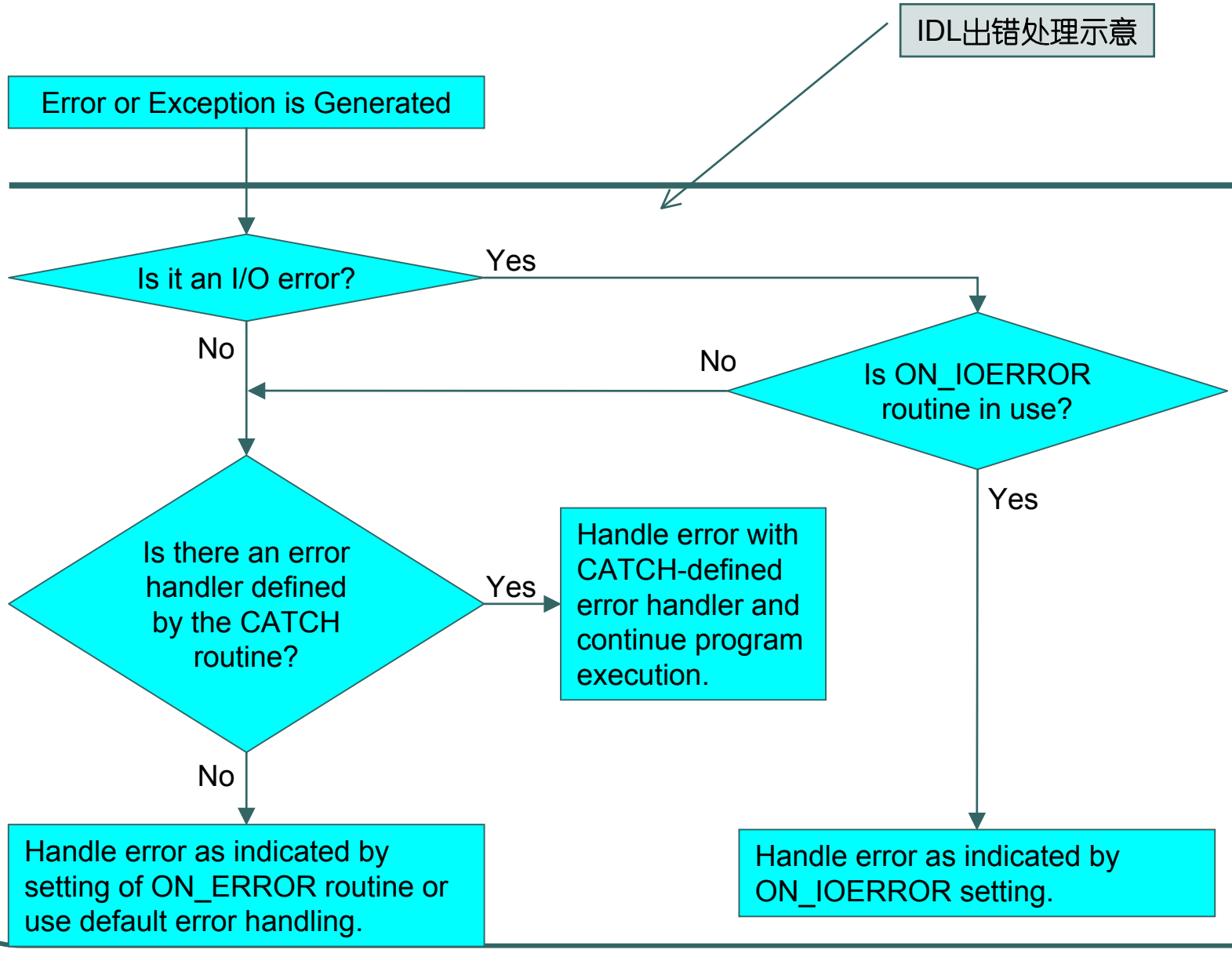
注意：使用on_ioerror ， null

on_error： 当程序运行出错时，并不执行一个新的语句，而是指明IDL应该怎样做。

值	行动
0	立即停止。缺省
1	立即停止，返回主程序
2	立即停止，返回程序调用模块
3	立即停止，返回程序模块

可以设置on_error ,1，或在命令行使用retall

catch： 格式：catch ,error_var。当程序执行到catch语句时，IDL为该模块记录一个错误处理语句，并将error_var赋值为0。若程序执行出错，则给error_var赋值相应的错误码，然后跳转到catch后第一条语句。注意：使用catch ,/cancel


IDL出错处理示意
Error or Exception is Generated
Is it an I/O error?
Yes
No
No
Is ON_IOERROR routine in use?
Yes
Is there an error handler defined by the CATCH routine?
Yes
Handle error with CATCH-defined error handler and continue program execution.
No
Handle error as indicated by setting of ON_ERROR routine or use default error handling.
Handle error as indicated by ON_IOERROR setting.

编译与运行

批处理：@bacthfile，运行
主程序：.run，编译、运行
过程和函数：.compile，编译；->proname，编译、运行。
编译规则：（1）编译到主程序后，编译停止
（2）编译到与文件同名的程序模块时，停止编译
（3）编译到文件末尾或适合其他规则时，停止编译
自动编译规则：当过程或函数出现在命令或代码中时，会自动被编译执行。
（1）过程或函数所在的文件应在当前工作路径和 !Path指定的路径中
（2）过程或函数名与文件名相同
编译函数：resolve_routine、resove_all。可用于程序模块中。
.sav：IDL->save，编译后存储为.sav文件，便于发布。但版本间不兼容。

输入输出
1. 常用概念

- 文件操作：

  openr，openw，openu，close

- 逻辑设备号：

  1～99，直接使用

  100～128，使用**get_lun**获取，**free_lun**释放

- 常用函数：

  dialog_pickfile，findfile，filepath

输入输出
2. 文本格式

自由格式：**readf，printf，strsplit**

readf中只接收变量引用，不接收值引用

format语法：**format='()'**，括号内为格式符及其组合

A：**[n]a[w]**，**n**为重复次数，**w**为输出宽度

I：**[n]i[w]**或**[n]l[w.m]**，缺省**w=7**，特殊用法：**i0**

F：**[n]f[w.d]**，缺省**w=15**

X：**[n]X**，空格

/：换行符

:：其后的格式不用于最后一项。如每个输出项后加一个‘，’时，最后一项不加。

C：**c()**，表示日期，接受**julian**日期。有丰富的子集

输入输出
3. 大型数据文件

- 二进制文件的关连变量处理

基本命令：**readu，writeu**

关联变量：大型重复单元二进制文件的有效读取手段，可以随机读取。

一个文件可建立多个关联，解决重复单元不一致的情况。

assoc()：

result=assoc(unit,array_structure [,offset] [,/packed])

输入输出
4. 常用格式（1）

ASCII_TEMPLATE	**Presents a GUI that generates a template defining an ASCII file format**
ASSOC	**Associates an array structure with a file**
BINARY_TEMPLATE	**Presents a GUI for interactively generating a template structure for use with READ_BINARY**
CDF Routines	**Common Data Format routines**
EOS Routines	**HDF-EOS (Hierarchical Data Format-Earth Observing System) routines**
HDF_BROWSER	**Opens GUI to view contents of HDF, HDF-EOS, or NetCDF file**
HDF_READ	**Extracts HDF, HDF-EOS, and NetCDF data and metadata into an output structure**
IDLffDICOM	**Contains the data for one or more images embedded in a DICOM part 10 file**
IDLffDXF	**Object that contains geometry, connectivity, and attributes for graphics primitives**
IDLffShape	**Contains geometry, connectivity and attributes for primitives accessed from ESRI Shape files**
MPEG_OPEN	**Opens an MPEG sequence**
MPEG_SAVE	**Saves an MPEG sequence to a file**
NCDF Routines	**Network Common Data Format routines**
PRINT/PRINTF	**Writes formatted output to screen or file**
READ/READF	**Reads formatted input from keyboard or file**
READ_ASCII	**Reads data from an ASCII file**
READ_BINARY	**Reads the contents of a binary file using a passed template or basic command line keywords**
READ_BMP	**Reads Microsoft Windows bitmap file (.BMP)**
READ_DICOM	**Reads an image from a DICOM file**
READ_IMAGE	**Reads the image contents of a file and returns the image in an IDL variable**
READ_INTERFILE	**Reads Interfile (v3.3) file**
READ_JPEG	**Reads JPEG file**
READ_PICT	**Reads Macintosh PICT (version 2) bitmap file**
READ_PNG	**Reads Portable Network Graphics (PNG) file**
READ_PPM	**Reads PGM (gray scale) or PPM (portable pixmap for color) file**
READ_SRF	**Reads Sun Raster Format file**

输入输出
5. 常用格式（2）

READ_SYLK	**Reads Symbolic Link format spreadsheet file**
READ_TIFF	**Reads TIFF format file**
READ_WAV	**Reads the audio stream from the named .WAV file**
READ_WAVE	**Reads Wavefront Advanced Visualizer file**
READ_X11_BITMAP	**Reads X11 bitmap file**
READ_XWD	**Reads X Windows Dump file**
READS	**Reads formatted input from a string variable**
READU	**Reads unformatted binary data from a file**
SOCKET	**Opens a client-side TCP/IP Internet socket as an IDL file unit**
TAPRD	**Reads the next record on a tape**
TAPWRT	**Writes data to a tape**
WRITE_BMP	**Writes Microsoft Windows Version 3 device independent bitmap file (.BMP)**
WRITE_IMAGE	**Writes an image and its color table vectors, if any, to a file of a specified type**
WRITE_JPEG	**Writes JPEG file**
WRITE_NRIF	**Writes NCAR Raster Interchange Format raster file**
WRITE_PICT	**Writes Macintosh PICT (version 2) bitmap file**
WRITE_PNG	**Writes Portable Network Graphics (PNG) file**
WRITE_PPM	**Writes PPM (TrueColor) or PGM (gray scale) file**
WRITE_SRF	**Writes Sun Raster File (SRF)**
WRITE_SYLK	**Writes SYLK (Symbolic Link) spreadsheet file**
WRITE_TIFF	**Writes TIFF file with 1 to 3 channels**
WRITE_WAV	**Writes the audio stream to the named .WAV file**
WRITE_WAVE	**Writes Wavefront Advanced Visualizer (.WAV) file**
WRITEU	**Writes unformatted binary data to a file**

常用控制语句

- Begin – End
- If – Then – Else
- For – Do
- While – Do
- Repeat – Until
- Case X of – else – endcase
- Expr? Expr1:expr2 (条件判断语句)

常用控制语句
1. If-Then-ELse

- IF (num GET 10) THEN index=2 ELSE index=4
- IDL>IF (num GET 10) THEN BEGIN $

  index=2 &$

  num=0&$

  endif else then begin $

  index=4&$

  endelse
- IF (num GET 10) THEN BEGIN

  index=2

  endif else begin

  index=4

  endelse

常用控制语句
2. For-Do

- For i=1,10 do a=a*i*2
- A=1

  For j=1, 10 Do BEGIN

  print, j

  A = a*j*2

  endfor

常用控制语句
3. While-Do

- i=0

  WHILE (i EQ 1) DO PRINT, i
- While (number LT 100) Do Begin

  index = amount * 10

  number = number+index

  endwhile

常用控制语句

4. Repeat-Until

- Repeat index=amount*10 until index GT 10

- Repeat Begin

  index=amount *10

  number=number + index

  Endrep until number GT 10

常用控制语句

5. Case

```
x=2
catch x OF
        1: PRINT, 'one'
        2: PRINT,'two'
        3: PRINT, 'three'
        4: PRINT, 'four'
ENDcatch

IDL prints:
two
```

常用控制语句
6. Switch

```
x=2
SWITCH x OF
        1: PRINT, 'one'
        2: PRINT,'two'
        3: PRINT, 'three'
        4: PRINT, 'four'
ENDSWITCH
```

IDL prints:
two
three
four

常用控制语句
7. 条件表达式与Goto

Expr? Expr1:expr2

```
index=(num ge 10)? 2:4
```

GOTO

```
For j = 0,n Do begin
    index=j * !pi* 5.165
    if index GT 3000 then GOTO, jump out
endfor
jump out: print, index
```

IDL——ENVI

- IDL中ENVI命令调用

```
envi, /restore_base_save_files
envi_init, /batch_mode
```