A Mugs Guide: **Debugging IDL**

Last Updated: April 12, 2005

Contents

1 There is no debugger

Although there isn't really a debugger in the the same sense as there is for compiled languages (gdb or dbx for fortran, c++, etc) all is not lost. There are a number techniques that can be employed to assist in the debugging of IDL. This "Mugs Guide" provides a brief overview of these techniques. See the IDL reference guides for detailed descriptions of the commands mentioned here.

2 Syntax highlighting

In IDL there are a large number of keywords, built-in functions, etc. To avoid potential conflicts/ambiguities you should avoid using any of these for your own variable/function/procedure names. A good syntax highlighter can help you to do this without needing to know the entire IDL language. Syntax highlighting also helps to make many simple typographical errors stand out, and as such can save a lot of time. Most modern text editors support syntax highlighting, however the quality (how much of the language is represented accurately) can vary. For the Nedit text editor, an IDL syntax highlighting pattern is available in the theory code repository.

3 Built into IDL

While there is no debugger for IDL there are a number of built-in commands and system variables which facilitate the debugging of IDL scripts, procedures and functions.

3.1 Useful Commands

STOP *The STOP procedure stops the execution of a running program or batch file. Control reverts to interactive mode.*

BREAKPOINT *The BREAKPOINT procedure allows you to insert and remove breakpoints in programs for debugging.* A key benefit of this procedure over STOP is that you don't need to edit the source code. For the execution of a routine contained in 'filename.pro' to stop on line 13 simply type:

```
BREAKPOINT, /set, 'filename.pro', 13
```

at the IDL command line before running the the routine which uses 'filename.pro'. Note 'filename.pro' needs to include a path if it isn't in the current directory.

HELP *The HELP procedure gives the user information on many aspects of the current IDL session.* Typing help with no arguments or keywords lists most of the useful information about procedure/function nesting and the variables at the current program level.

PRINT *Prints to standard-out (terminal) the comma separated list of constants, variables and expressions(a+b, etc) that follow it.* Generally it is more useful to display variables for debugging purposes by using "help", as it automagically adds the variable name and type. Print is however useful if you want to look at the content of arrays (as help simply tells you the size of an array)

```
PRINT, 'Array:', Array
```

or if you are simply keeping track of where you are in your code

```
PRINT, 'In loop over position, iteration:  ', i
```

where i is the loop counter, eg in a for loop.

MESSAGE *Issues error and informational messages using the same mechanism employed by built-in IDL routines.* In its simplest form this can be used in place of a print statement, it has the advantage of displaying traceback(name of current routine and the routines through which this routine was reached, also the full path to the source file and the line within the source file) information by default.

Plotting Often a quick way to localise the source of a bug where arrays of numbers are involved is to plot the arrays. IDLs itools (iplot, icontour, isurface, ivolume) are good for this as they enable interactive(mouse point and click) zooming, rotating, etc. Also useful is directly displaying 2D arrays as an image (tvscl). This avoids the potential smoothing out, and thus hiding fine scale problems, which can occur with procedures like contour.

3.2 Executive Commands

.STEP *Executes a single statement.*

.CONTINUE *Continues execution of a stopped program.*

.TRACE *Similar to .CONTINUE, but displays each line of code before execution.*

.RETURN *Continues execution until encountering a RETURN statement.*

.COMPILE *Compiles text from files or keyboard without executing.* This is useful when you've made changes to a procedure or function. Procedures and functions are compiled the first time they are needed and simply used after that, so changing the source code without recompiling won't affect what IDL is running.

IDL "remembers" things that you have done in the current session, so it's possible for bugs from earlier in a session to continue to cause problems. There are two executive commands which are intended for cleaning up changes you've made:

.RESET_SESSION *Resets much of the state of an IDL session without requiring the user to exit and restart the IDL session.*

.FULL_RESET_SESSION *Does everything .RESET_SESSION does, plus additional reset tasks such as unloading sharable libraries.*

3.3 Relevant system variables

!EXCEPT The default setting for this system variable collects up mathematical exceptions and displays them at the end of a run. It can be useful to have these messages printed as they occur, this is achieved by setting the !EXCEPT system variable to 2. At the IDL prompt, or from within an IDL script, procedure, or function type:

```
!EXCEPT=2
```

!PATH This system variable contains the directory path through which IDL searches for files containing functions and procedures. The path to the directories containing your procedures or functions need to be in this !PATH variable for IDL to find them.

!WARN Setting the values of this structure to byte value 1 causes warning messages to be issued when obsolete routines and system variables are being used, also warns of the use of parentheses when specifying an array index.

```
!warn={!warn, 1B, 1B, 1B}
```

4 A Note on Speed Issues

While not really bugs, order of magnitude decreases in the speed of IDL code can occur due to simple things such as the incorrect ordering of loops, and the

use of loops and if-blocks instead of array operations and where statements. So it is worth doing things the "right way"...
For a brief overview of some of the key speed issues in IDL see "Tips & Tricks for Efficient IDL Programming" at

http://www.rsinc.com/services/techtip.asp?ttid=1799

which is part of the RSI (the company that owns IDL) tech support web pages.
Also, for a more detailed discussion, see the section "Writing Efficient IDL Programs" in the "Building IDL Applications" manual.

5 Further Information

For electronic versions of the documentation (linked searchable .pdf), at the IDL command line type: ?

Useful Web Pages:

http://groups-beta.google.com/group/comp.lang.idl-pvwave
http://www.dfanning.com/
http://www.rsinc.com/ (look under support)
http://www.astro.washington.edu/deutsch/idl/index.html
http://cow.physics.wisc.edu/~craigm/idl/idl.html