Subject: IDL excels in debugging??? Do you know something I dont?
Posted by Russ Welti on Thu, 18 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Wally Gross wrote:

>It's nice to know that a well written IDL program can be faster than
>the corresponding C program, but on the projects I've been on speed
>wasn't the most important concern. Getting the system developed
>and debugged was always the biggest problem. In that area, IDL really
>excels though systems like Turbo C for the PC are terrific for
>development and debugging also. IMHO the real challenge with IDL is
>creating maintainable code.

I agree about the maintainable code problem. Being interpreted (IMHO)
tends to make any language at risk for quick and dirty programming --
prototypes and experimental code changes that "never got documented".
But then we chose IDL *because* we wanted rapid prototyping capabilities ;)

Of course I blame myself if I fall into bad habits, but IDL is not very
consistent in its overall design and methods for accomplishing various
programming tasks. For example, there are multiple ways to accomplish
several fundamental operations, sometimes undocumented, and without
the kind of overall simplicity of concept that characterizes, for example,
object-oriented paradigms. But IDL *evolved*; it was not "designed".

In IDL, online help is really a must, because the lack of consistency
means one must constantly refer to each procedure/function to remember
how that particular creature works... Early in my exposure to IDL I read
here that "IDL is a hacker's language". I have often reflected on that...

I would REALLY take exception to the statement that IDL is debuggable.
If you have used a good Unix debugger, it is hard to even compare the total
lack of debugging tools (usable anyway) in IDL. You may notice this is a
hot button for me. Breakpoints are a JOKE. Even perl has a very useful,
line-oriented debugger which RSI should use as a model, I believe.
idltool is an admirable attempt, but unwieldy and unreliable.

The most useful debugging technique (other than the good ole PRINT statement)
I know of is the following 2 line routine, offered to me once by
rep2857@sbsun0010.sbrc.hac.com (Mike Schienle)

```
; BREAK.PRO: a "debugging" routine. it always causes an error. Period.
; A call to 'break' in IDL will break IDL and return to the routine
; which called it, allowing you to examine all variables' values at
; the point it was called.   There is generally no way to continue execution,
; you must "RETALL & XMANAGER" (aargh!).    R. Welti; from M.Schienle
```

PRO
END

In fact, I would love to read a discussion of what other people are using for debugging techniques / tools.

```
                                        /
Russ Welti                                  /-\
                                      (c-g)
University of Washington                          \-/
Molecular Biotechnology                            /
PO Box 352145                                   /-\
Seattle, WA  98195                              (a-t)
rwelti@u.washington.edu                            \-/
(206) 685 3840 voice  (206) 685 7344 FAX            /
http://chroma.mbt.washington.edu/graphics/gif/russ.gif
```