Subject: Re: IDL as programming language?
Posted by stl on Thu, 01 Sep 1994 07:03:58 GMT
View Forum Message <> Reply to Message

I just have to give a little plug for IDL...

In article <1994Aug30.235145.15667@henson.cc.wwu.edu> n9140397@gonzo.cc.wwu.edu (Michael Hamilton) writes:
> In article <1994Aug28.122441.32497@waikato.ac.nz> abz@waikato.ac.nz writes:
>
>> (i) Accuracy. Our current version of IDL seems to prefer doing calculations
>> in single precision, while we prefer double. Has this been improved in the
>> latest version? (e.g. in our current version, routines like LUDCMP work in
>> s.p., despite being passed d.p. arguments.)
>
IDL allows types to be Dynamic. Double precision is not a problem, you
just must make sure that the variables you are dealing with at the time
are all DOUBLES. One of the nicest features with IDL is that you can
interact with variables of different types, and generally not worry
about them, almost no other language allows you to do this (low level
language at least) Therefore, if you initially define all your
variables as DOUBLES, you should have NO problem.
>
>> (ii) Speed. Some of us Grads are running some really time consuming programs
>> (large arrays, large loops). How does IDL compare with (say) FORTRAN in
>> general, speedwise? (my impression is that it's pretty slow, but I could be
>> wrong...)
>
The responses to this are extremely interesting. When programmed
correctly (ie: a few loops as possible, accessing memory in the correct
order, etc) IDL SCREAMS. ANd the speed is obtained with very little
code. When dealing with matrix calculations, I have hardly heard anyone
complain about its speed. GRANTED, if you do not use the built in
operators, use lots of loops, and program as if you have to handle all
the indexing,.. then its pretty slow..
>
>> (iii) Memory. How does IDL's memory management compare? Again, some of our
>> programs (FORTRAN) have a tendency to gobblelarge chunks of memory (probably
>> bad programming, but still...)
>
No idea really. (probably not even close to C, maybe to Fortran but I
really have no idea) But, hey, memory is cheap... (usually)
>
>> (iv) What is a large IDL code like to debug?
>
> UGH! Again, my opinion. Some people will probably like debugging idl code
> over FORTRAN code (or some other legitamate language). But, I would venture
> that they do not know how to use a real debugger like dbx...

HUM, hate to respond against someone elses answer, but I disagree here.
As for DBX, well.. no comment. If your code is programmed Modularly,
ie: one routine per file, I find it very nice to debug because of the
following:
 -IDL tells you the exact line the error occured (I hear that the PC
 and MAc version even thoughs you into the file to debug it.. NICE)
 -The error messages usually are pretty meaningfull
 -When an error does occure, you are left in that environment, and
 can test any variable you want (no need to start a debugger, and set
 all variables you want to look at, etc)
 -You can even find the problem, fix it, recompile the code and
 continue where you die before. Or you can try out new ways to do
 what your code does in an interactive mode.

Granted, it has no GUI oriented interface for a debugger, but I think
its easier then debugging compiler/linker/memory and other problems that
you might encounter in C. Its A a higher level language then C and
fortran, so its pretty tough to compare them.

>
>> (v) How 'robust' is IDL as a programming language? We have a variety of
>> different programming styles here -- some prefer 'quick and dirty' programmin
>> others a more structured approach. Forgive my possible ignorance, but I have
>> the impression that IDL as a language is more suited to the 'quick and dirty'
>> approach. Is this true? Does IDL as a programming language have many
>> glitches or inconveniences from a mathematical programmers point of view?
>
Okay, this is a critical question. Normally I find it VERY stable. But
I am learning that when working with IDL on a network with different
versions of UNIX (Solaris and SUNOS) and apparently therefore different
X servers, it gets somewhat flaky.

Another thing worth thinking about is building huge applications with
IDL. It CAN be done. And more and more easily with every release, but
you need to plan, and define standards just like any other language. It
does tend to be a quick and dirty langauge, but MODULAR code is
possible. Just be neat, define a regular tree structure to code, put
each routine in a seperate file, define the paths in the startup file,
etc.. I have been doing it for numerous big projects. Also with the
introduction of WIdgets, interfaces are possible, and are standard
across platforms (somewhat). But don't expect control over them as if
you were prograsmming from Motif, but they are really not bad.

Finally, yes, there is no possible way to build a runtime of code
without an IDL license to run it. Kind of a bummer, but unterstandable
when you are marketing a product, and most other higher level langauges
are like this.. But you can compile the code, and save it for execution
later, this works fine.

Also, now you can call IDL functions via RPC from someother langauge.
ANd the conectablitlity to C and Fortran LIbraries is pretty good. Its
still growing, and really is an amazing analysis tool first and formost!
But don't expect that just because its easy to use that an
nonexperienced programmer can build a huge application in it just
because its a relatively easy langauge..

Well thats
> My 2 cents....

 -stephen
--
Stephen C Strebel / SKI & TELE TO DIE
strebel@sma.ch / and
Swiss Meteorological Institute, Zuerich / LIVE TO TELL ABOUT IT
01 256 93 85 / (and pray for snow)
