Subject: Re: Discussion on global variables in IDL
Posted by Martin Schultz on Wed, 01 Aug 2001 12:50:25 GMT
View Forum Message <> Reply to Message

alt@iszf.irk.ru (Altyntsev Dmitriy) writes:

> Hello,
>
> I'd like to discuss global variable management in IDL. I've read a
> pair of NG threads on that theme, and would like to add some
> consideration and share experience. Some time ago I've sent a letter
> to RSI with some proposals, but since then two version have been
> released and I don't see any movement in the direction of improving
> global vars management. It seems that everyone is satisfied. Is it
> really so?

Well, I don't think the situation is as bad as you paint it. First of all
I recommend objects - which are in fact "global" (as long as you don't loose
the object reference), secondly widgets (which provide you with a UValue
field and thu sgive you an opportunity to assess the underlying widget object
reference as long as you see it on the screen ;-) [but, seriously: don't
tell David that you have seriously even thought about using a COMMON block
in a widget program; he will slam you with a box of his books]. Finally,
you have user variables (the ! thingies). So, if you like, you can make
up your own global variable system by defining a system variable !GLOBAL
which contains a pointer pointing to a pointer (or object reference) array;
plus, you will need some "management" information. Come to think of it;
the behaviour you would like to see there, is exactly what a container
object is designed for. And if you use my version (instead of IDL_Container),
you can even search your variables by name.

Cheers,

Martin

--
[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[

Subject: Re: Discussion on global variables in IDL

Posted by Pavel A. Romashkin on Wed, 01 Aug 2001 17:57:20 GMT
View Forum Message <> Reply to Message

I think Martin went more into the programming details here, giving
useful development ideas. What I would like to address, is the *purpose*
of IDL sessions.
You can either use IDL for ad-hoc testing of ideas (and this is what it
looks to me Dmitry is doing, from the "write two lines, debug, and write
next two"). Or you can write an *application* that will process the data
you need. In the latter case, if it turns out the user needs "global"
variables, it means that the application is not encompassing enough
within the IDL session. It means that an *application* is trying to
access ad-hoc data. If you write a program in IDL, think about it as if
it is to become a standalone program: how do you pass "global" data to
Photoshop, for example? And what is "global" data in this case?
If you do need a widget program to aid in the ad-hoc work, pass in the
"global" data via parameters or keywords.
Of course there are special cases, for instance data aquisition systems.
But if you are at that level and still remember about common blocks, you
are risking a dishonorable discharge from the EPA :-)
Common blocks are useful - there is little doubt about it. But I can see
very, very few instances when they are the only way. And the entire idea
of "global" variables seems to me to be resulting from the evolutionary
approach to programming - start with command line data analyses, get
tired of copy-paste. write code, expand it and still need to access the
data the way you started it.
Cheers,
Pavel
P.S. Of course, all the above is pure speculation. But I learned the
hard way that, no matter how tempting it is to just sit and write IDL
code as you think, it pays to design your data organization first, and
design the ways it is to be processed. Even more so when trying to write
useful object programs.

Subject: Re: Discussion on global variables in IDL
Posted by Martin Schultz on Wed, 08 Aug 2001 10:55:17 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" <pavel.romashkin@noaa.gov> writes:

> ...
> Common blocks are useful - there is little doubt about it. But I can see
> very, very few instances when they are the only way. And the entire idea
> of "global" variables seems to me to be resulting from the evolutionary
> approach to programming - start with command line data analyses, get
> tired of copy-paste. write code, expand it and still need to access the
> data the way you started it.

And this is exactly their weakness. I would guess that most IDL applications
are born out of ad-hoc code once written to analyse a specific situation,
and then generalized. To get rid of common blocks in this provess is much
harder than to avoid them in the first place.

Cheers,

Martin

```
--
[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[ [[[[[[[
[[ Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie    [[
[[                      Bundesstr. 55, 20146 Hamburg             [[
[[                      phone: +49 40 41173-308                  [[
[[                      fax:   +49 40 41173-298                  [[
[[ martin.schultz@dkrz.de                                        [[
[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[ [[[[[[[
```

Subject: Re: Discussion on global variables in IDL
Posted by alt on Fri, 10 Aug 2001 09:32:46 GMT

View Forum Message <> Reply to Message

Martin Schultz <martin.schultz@dkrz.de> wrote in message
news:<ylw66c8q6zy.fsf@faxaelven.dkrz.de>...
> alt@iszf.irk.ru (Altyntsev Dmitriy) writes:
>
>> Hello,
>>
>> I'd like to discuss global variable management in IDL. I've read a
>> pair of NG threads on that theme, and would like to add some
>> consideration and share experience. Some time ago I've sent a letter
>> to RSI with some proposals, but since then two version have been
>> released and I don't see any movement in the direction of improving
>> global vars management. It seems that everyone is satisfied. Is it
>> really so?
>
> Well, I don't think the situation is as bad as you paint it.
> First of all I recommend objects
> Cheers,
>
> Martin

My first reaction on yours answers was that I had been misunderstood.
I even post a message about it, but GOOGLE did not put it for some
reason in this thread. Then I started to analyze what I need and what
OOP offers me. I have read a pair of OOP books, scanned one more time

this NG as far as possible, looked up Martin's IDL_Container and
others. And to my great surprise discover that I have reinvented the
OOP wheel. It is funny (or sadly?) that I am using now procedure
naming conventions in the style of OOP like OBJ_Method. And I am using
pointer to data of my object in a style I have described before. I was
completely not right to name this problem as GLOBAL variable problem.
Global is global. But it is a problem how to share some data between
several procedures. And it actually is a problem of OOP implementing
in IDL. And having looked up through this NG I have found that this is
a problem in IDL and many people are NOT satisfied of current IDL OOP
state. Some of them have written their own 'containers', others do not
use OOP, and some even threatened with stopping to use IDL.

The most interesting NG thread (IMHO) was "Top 10 wishes". (BTW, I
could not find the end and the result of this epic work.) I wish this
column were persistent and wish to describe my current wish to IDL
OOP.

So, what is the most annoying thing with IDL OOP for me? The fact that
when I describe class I loose freedom of changing field type and size.
I can not have undefined field in class too. And I can not add field
during execution. Let us view very simple typical example.

```
IND = WHERE(A GT 0) ; the essence of IDL style
```

Suppose we want this IND as field of some class. Then I need (as David
tired to repeat) to use pointers and write like this:

```
function class1::INIT
 self.ind = ptr_new(/all)
 return, 1
end
pro class1::CLEANUP
 ptr_free, self.ind
end
pro class1::method1
 ...
 *self.ind = where(A GT 0)
end
pro p
 struct = { class1, ..., IND:ptr_new(), ... }
 Obj1 = obj_new( 'class1' )
 Obj1->method1
 obj_destroy, Obj1
end
```

So, instead of one simple string I must initialize and destroy pointer
manually. If during development I suddenly wanted to have IND as field

of the class, I must add it in class definition, in class init, and in
class cleanup. Or use some slow container, because all of them are
written in IDL with a lot of operations.

I do not understand why IDL can not do all this stuff itself. It can
be done several ways. As for me, I would like to form class fields
during execution. It seems to me on the whole of IDL style.

```
pro class1::method1
 ...
 self.IND = where(A GT 0)
 ...
 self.IND = 'We can change type of the field'
 ...
 tmp = temporary(self.IND) ; or destroy it
end
pro p
 struct = { class1, public ..., IND, ..., privite ... }
 Obj1 = obj_new( 'class1' )
 Obj1->method1
 obj_destroy, Obj1
end
```

As soon as IDL meet self.IND it inits it at once. And destroy by
obj_destroy or reassigning like local variables.

I think that it would be fine to have an opportunity even not define
variable in class description but it involves complicated symbol
tables managing by IDL during compile time or slow during execution.
It may be difficult task but solvable.

And of course, I must have an opportunity to assign class method as
event handler in widget applications.

What would you say?

The other my "Top 10" very shortly
1) Keystroke events and MDI interface (ENVI and others apps might be
much more convenient). I use widget_text trick now.
2) Handy and effective tools for viewing images and plots (colors,
multichannel, cursor position, zoom, image enhancement, export,
printing, ...). I have my own tools. They are far from perfection.
3) Class explorer, so I could quickly look up my fields and methods.
Now I use external text editor search capabilities.
4) GIS functions to work with vector data (polygon intersections,
area, distance ...) Writing myself.
5) FTP object
6) Low or null prize of run-time package

This will be enough.

Regards,
Altyntsev Dmitriy