Subject: Re: Commons, Was: can i place a job advert
Posted by Adam Rankin on Mon, 26 Nov 2001 19:16:58 GMT
View Forum Message <> Reply to Message

Well, I'm probably the youngest IDL programmer here and for me, Common
Blocks represent ease of use and simplicity. Granted I don't do near the
difficulty or complexity of programming that others do however I am still
a programmer and for what I do, common blocks work and they have cause me
no problems yet.

I'm sure once I develop real skill and get into the "normal" level of
coding that common blocks will rise up to nip me in the gludius minimus
but until them I say "Yay Common Blocks!"

My 2 canadian cents, worth approximately 0.0025 cents american.

-Adam

Subject: Re: Commons, Was: can i place a job advert
Posted by Paul van Delst on Mon, 26 Nov 2001 19:55:25 GMT
View Forum Message <> Reply to Message

Richard Younger wrote:
>
> Paul van Delst wrote:
>>
>> Well, my comment was meant to try to get those that swear profusely
>> that common blocks are a Bad Thing to explain why. I place common
>> blocks in the same category as GOTO statements - if used incorrectly
>> they can lead to unbelievably bad source code that may or may not
>> work.
>
> Well, the snooty Computer Science answer to this is that common blocks
> or global data completely separate the context of the data from its
> content. Similarly, with GOTOs, it's easy to separate the context of
> one snippet of code from another. When you have a global variable, you
> (the indefinite you, which could actually include someone else :-))
> have no real idea what sort of code is using that data. I think the
> idea of prohibiting them is to remove the chore and responsibility of
> keeping track of the context from anyone else (or any other process of
> yours) that wanders along. It's a Good Thing(tm) when the programming
> system you use discourages methodologies that tend to cause confusion
> and mistakes.
>
> Admittedly, common blocks have some features apart from purely global
> data that discourage errors. They have to be specifically invoked, can't
> be resized (upwards), and usually IDL projects aren't so big that one

> person can't keep track of a set of well controlled commons. The
> examples given why commons in IDL specifically are bad seem to involve
> doing multiple things at the same time, or with multiple copies of the
> same program running.

This sort of usage/limitation suggests (to me) a program design flaw in that a particular construct (common) was being used when another would do the job better. Lots of other code constructs have this "feature" too. I'm not disagreeing with you, just stating that there are sometimes when using a common block is called for (be it for clarity, simplicity, whatever) despite the "snooty Computer Science" viewpoint.

> For myself, I seem to get along nicely without commons.

Me too. Haven't needed 'em since Fortran 90 introduced modules and the ability to make whatever
I wanted to be a public or private entity of that.....oops, wrong newsgroup.

> Mind you, I'm not supporting banning commons as dogma, but I think there
> are enough general objections to them to ask people to think a bit
> before they rush out and use them everywhere they can.

The important part of the above sentence is the fragment: "...ask people to think a bit..."
:o)

I particularly liked the sentence:

"It's a Good Thing(tm) when the programming system you use discourages methodologies that tend
to cause confusion
and mistakes."

If this is to be believed, pointers would never have been invented :o)

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Commons, Was: can i place a job advert
Posted by Pavel A. Romashkin on Mon, 26 Nov 2001 20:35:14 GMT
View Forum Message <> Reply to Message

This topic is irresistable.
How about we issue a Challenge:

Please modify the object definition below or create a method for objects
to be aware of each other after creation:

```
;**********
pro Test__define
result = {'TEST', data_holder : ptr_new(), inherits 'IDLgrModel'}
end

;so that if

a = obj_new('TEST')
b = obj_new('TEST')

;then it is possible to do something like

b_CTM_matrix = a -> Get_others_CTM(a_keyword = '?')

; without passing B to the call?
;**********
```

Submissions using bulk heap searches and .sav files are not allowed :)

Cheers,
Pavel

Richard Younger wrote:
>
> Mind you, I'm not supporting banning commons as dogma, but I think there
> are enough general objections to them to ask people to think a bit
> before they rush out and use them everywhere they can.

Subject: Re: Commons, Was: can i place a job advert
Posted by Martin Downing on Tue, 27 Nov 2001 00:20:19 GMT

View Forum Message <> Reply to Message

Well I've removed all the graphics stuff -I havent got a clue what that was
about and you can add what you like later!
My solution is to link all objects as a singly linked list which has its
head pointer (named objTop in this code)
stored using a COMMON BLOCK so that it is visible to all object instances.
This could be considered a static object
member instance, rather like you can find in some C++ Classes. In C++ these
beasts have only one value but are visible from all instances of the class -
incredibly useful for stacks.
Ok so strictly the variable is not private to the class, but it takes
someone daft enough to copy the
COMMON identifier: "__TEST_PRIVATE_STATIC_MEMBERS" elsewhere to access the
pointer!

I think this is a very reasonable use of common blocks, but I await to see alternatives.

cheers

Martin

When you run it you should get the following:

```
IDL> pavels_test
three
one
```

```
================================================
pro Test__define
result = {TEST, data_holder : ptr_new(),objPrevious:obj_new()};, inherits
'IDLgrModel'}
end

function TEST::init, data=data
 COMMON __TEST_PRIVATE_STATIC_MEMBERS, objTop

 ; if undefined, point objTop to a NULL object
 if n_elements(objTop) eq 0 then objTop = obj_new()

 self.objPrevious = objTop
 objTop = self

 ; add the rest of your initialisations
 self.data_holder = ptr_new(data)

 return, 1
end



function TEST::GET_OTHERS, index ;FromTop
 COMMON __TEST_PRIVATE_STATIC_MEMBERS, objTop

 obj=objTop
 i = 1
 while i LT index do begin
     if obj_valid(obj) then obj = obj.objPrevious
  if obj ne self then i = i+1
 endwhile
 if obj_valid(obj) then data = *(obj.data_holder) else data = 0

 return, data
end

; delete a single instance
pro TEST::CleanUp
 COMMON __TEST_PRIVATE_STATIC_MEMBERS, objTop

 ; free data
 ptr_free, self.data_holder

 ; relink object pointers
 if self eq objTop then begin
  objTop = self.objPrevious
 endif else begin
  obj=objTop
  while obj ne self do begin
   help, obj
   next = obj
      obj = obj.objPrevious
  endwhile
  next.objPrevious = self.objPrevious
 endelse
end

; delete all instances of this object class
pro TEST::CleanUpAll
 COMMON __TEST_PRIVATE_STATIC_MEMBERS, objTop

 obj=objTop
 while obj_valid(obj) do begin
     objPrev = obj.objPrevious
  obj_destroy, obj
  obj = objPrev
 endwhile
end


pro pavels_test

a = obj_new('TEST', data = "one")
b = obj_new('TEST', data = "two")
c = obj_new('TEST', data = "three")

;then it is possible to do something like
print, b->get_others(1)
print, b->get_others(2)
```

c->CleanUpAll

end

 == =

--

Martin Downing,
Clinical Research Physicist,
Grampian Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.
Tel. 01224 556055 / 07903901612
Fax. 01224 556662

m.downing@abdn.ac.uk

"Pavel A. Romashkin" <pavel.romashkin@noaa.gov> wrote in message
news:3C02996F.ABD6D1D8@noaa.gov...
> This topic is irresistable.
> How about we issue a Challenge:
>
> Please modify the object definition below or create a method for objects
> to be aware of each other after creation:
> ;**********
> pro Test__define
> result = {'TEST', data_holder : ptr_new(), inherits 'IDLgrModel'}
> end
>
> ;so that if
>
> a = obj_new('TEST')
> b = obj_new('TEST')
>
> ;then it is possible to do something like
>
> b_CTM_matrix = a -> Get_others_CTM(a_keyword = '?')
>
> ; without passing B to the call?
> ;**********
> Submissions using bulk heap searches and .sav files are not allowed :)
>
> Cheers,
> Pavel
>
> Richard Younger wrote:

>>
>> Mind you, I'm not supporting banning commons as dogma, but I think there
>> are enough general objections to them to ask people to think a bit
>> before they rush out and use them everywhere they can.

Subject: Re: Commons, Was: can i place a job advert
Posted by Richard Younger on Tue, 27 Nov 2001 01:17:28 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> This topic is irresistable.
> How about we issue a Challenge:
>
> Please modify the object definition below or create a method for objects
> to be aware of each other after creation:

Let the testing begin!

Well, I can do it with a system variable (see below) but this has all of
the disadvantages of the common block, including namespace issues.
And the namespacing discussion has already been done. I could also do
it with a few lines inserted at the beginning of the startup file,
rushing to gain the ObjHeapVar1 space first. I could also conceive of
making a DLM to declare and reference a global C variable.

But you've specifically said that there can be no direct context
communication between objects. The problem is framed so that one needs
some information on a global level. Really, if you're going to
subscribe to objects fully, you shouldn't need to find other objects
without referencing them.

The question is what are you doing that truly requires global
information? Besides, um, main level interactivity? And besides, er,
main level widget interaction? And, er, um, *mumble*
hardware-dependendent stuff, like *mumble* display controls. Okay, so
maybe there occasionally are good reasons to need some global
information. :-)

Sorry for sounding grumpy, and I agree that commons have their place,
I'd just thought I'd chime in with something a little more reasonable
than "Commons are bad."

Best,
Rich

--

Richard Younger

```
FUNCTION Test::init

 CATCH, error
 IF error NE 0 THEN BEGIN
  catch, /cancel
  table = OBJ_New('IDL_container')
  DEFSYSV, '!RYtable', table
 ENDIF

 table = !RYtable
 table->add, self

 RETURN, 1

END

PRO Test::cleanup

 table = !RYtable
 table->remove, self
 IF table->count() EQ 0 THEN BEGIN
  Obj_destroy, table
  !RYtable = OBJ_NEW()
 ENDIF

 self->IDLgrModel::cleanup

END

FUNCTION Test::get_others

 table = !RYtable
 RETURN, table->Get(/all)

END

PRO Test__define
 self = {TEST, $
  data_holder  : ptr_new()  , $
  inherits   IDLgrModel   $
 }
END
```

Subject: Re: Commons, Was: can i place a job advert
Posted by Richard Younger on Tue, 27 Nov 2001 02:44:23 GMT
View Forum Message <> Reply to Message

It occurs to me that I neglected some inheritance.

There. I feel better. Still, no warranties apply.

Rich

--
Richard Younger

```
FUNCTION Test::init, _Extra=extr

   status = self->IDLgrModel::init(_Extra=extr)

   CATCH, error
   IF error NE 0 THEN BEGIN
      catch, /cancel
      table = Obj_New('IDL_container')
      DEFSYSV, '!RYtable', table
   ENDIF

   IF status EQ 1 THEN BEGIN
      table = !RYtable
    table->add, self
   ENDIF

   RETURN, status

END

PRO Test::cleanup

   table = !RYtable
   table->remove, self
   IF table->count() EQ 0 THEN BEGIN
      Obj_destroy, table
      !RYtable = OBJ_NEW()
   ENDIF

   self->IDLgrModel::cleanup

END

FUNCTION Test::get_others, _Extra=extr
```

```
   table = !RYtable
   RETURN, table->Get(/all, _Extra=extr)

END

PRO Test__define
   self = {TEST, $
      data_holder    : ptr_new()    , $
      inherits       IDLgrModel     $
   }
END
```

Subject: Re: Commons, Was: can i place a job advert
Posted by Richard Younger on Tue, 27 Nov 2001 02:49:26 GMT
View Forum Message <> Reply to Message

Hi, Paul.

Paul van Delst wrote:
>
>> Mind you, I'm not supporting banning commons as dogma, but I think
>> there are enough general objections to them to ask people to think a
>> bit before they rush out and use them everywhere they can.
>
> The important part of the above sentence is the fragment: "...ask people to think a bit..."
> :o)

Yup. I heard somewhere that mastery lies not in merely knowing the
rules, but in knowing just when to break them. :-)

> I particularly liked the sentence:
>
> "It's a Good Thing(tm) when the programming system you use discourages
> methodologies that tend to cause confusion and mistakes."
>
> If this is to be believed, pointers would never have been invented :o)

I would certainly discourage using pointers when you don't need them.
They tend to cause confusion and mistakes. :-)

But sometimes it's just too hard to do without, like sometimes you need
common blocks. Well, according to many here, anyway.

There's someone here at the lab who likes to tell of his Fortran IV
program where he declared a big global array at the beginning, and then
wrote his own interpreter to internally allocate and deallocate chunks
of that array. He just needed the dynamic memory, and there really

wasn't another way to do it.

I guess the practice of programming defies absolutes.

Best,
Rich

--
Richard Younger

Subject: Re: Commons, Was: can i place a job advert
Posted by Heike Koch-Beuttenmue on Tue, 27 Nov 2001 09:21:36 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> Well I've removed all the graphics stuff -I havent got a clue what that was
> about and you can add what you like later!
> My solution is to link all objects as a singly linked list which has its
> head pointer (named objTop in this code)
> stored using a COMMON BLOCK so that it is visible to all object instances.
> This could be considered a static object
> member instance, rather like you can find in some C++ Classes. In C++ these
> beasts have only one value but are visible from all instances of the class -
> incredibly useful for stacks.
> Ok so strictly the variable is not private to the class, but it takes
> someone daft enough to copy the
> COMMON identifier: "__TEST_PRIVATE_STATIC_MEMBERS" elsewhere to access the
> pointer!
> I think this is a very reasonable use of common blocks, but I await to see
> alternatives.
>
> cheers
>
> Martin
>
Can anybody explain me, why this mixture of using Common Blocks and
Objects is possible? (Fortran and C++?)

Heike

Subject: Re: Commons, Was: can i place a job advert
Posted by Martin Downing on Tue, 27 Nov 2001 10:45:11 GMT
View Forum Message <> Reply to Message

"Richard Younger" <younger@ll.mit.edu> wrote in message
news:3C02E9A8.F29A4C7B@ll.mit.edu...
> "Pavel A. Romashkin" wrote:
>>
>> This topic is irresistable.
>> How about we issue a Challenge:
>>
>> Please modify the object definition below or create a method for objects
>> to be aware of each other after creation:
>
> Let the testing begin!
>
> Well, I can do it with a system variable (see below) but this has all of
> the disadvantages of the common block, including namespace issues.

I would say system variables have far more disadvantages. Any system
variable can be listed and changed by the user and so is very much public.
whereas a common block can be hidden with such a criptic identifier that it
will only be accessible by the developers object code or to an outright
hacker! Used in that sense common blocks extend the power of IDL immensely
(IMHO).

Pinching some code from automaticaly generated C++ header file which needs a
unique identifier that will not clash with any other namespace:

COMMON _MY_OBJECT_COMMON__B32EE283_F153_11D4_8E3A_0080AD7D5B20__,
a_very_hidden_variable

who would manage to copy that!!

Martin

Subject: Re: Commons, Was: can i place a job advert
Posted by Pavel A. Romashkin on Tue, 27 Nov 2001 20:53:43 GMT
View Forum Message <> Reply to Message

Richard Younger wrote:
>
> The question is what are you doing that truly requires global
> information? Besides, um, main level interactivity? And besides, er,
> main level widget interaction? And, er, um, *mumble*
> hardware-dependent stuff, like *mumble* display controls. Okay, so
> maybe there occasionally are good reasons to need some global
> information. :-)

I think this sums it all up nicely. I personally also think that
Richard's listed basically all instances where truly global access is required.

And, at this point, we have submissions with four ways of implementing
truly global access:

1. Common block
2. System variable
3. Run-time function
4. File

Of these, Common blocks look the best to me.
Any other ideas?
Cheers,
Pavel

Subject: Re: Commons, Was: can i place a job advert
Posted by Dave Greenwood on Wed, 28 Nov 2001 15:57:44 GMT

View Forum Message <> Reply to Message

In a previous article, Richard Younger <younger@ll.mit.edu> wrote:
> "Pavel A. Romashkin" wrote:
>>
>> This topic is irresistable.
>> How about we issue a Challenge:
>>
>> Please modify the object definition below or create a method for objects
>> to be aware of each other after creation:
>
> Let the testing begin!
>
> Well, I can do it with a system variable (see below) but this has all of
> the disadvantages of the common block, including namespace issues.
> And the namespacing discussion has already been done. I could also do
> it with a few lines inserted at the beginning of the startup file,
> rushing to gain the ObjHeapVar1 space first. I could also conceive of
> making a DLM to declare and reference a global C variable.
[snip]

Out of curiousity - what does gaining "the ObjHeapVar1 space first" do
for you? How would you use that for something useful?

Thanks,
Dave

Dave Greenwood Email: Greenwoodde@ORNL.GOV
Oak Ridge National Lab %STD-W-DISCLAIMER, I only speak for myself

---

Subject: Re: Commons, Was: can i place a job advert
Posted by Pavel A. Romashkin on Wed, 28 Nov 2001 16:57:24 GMT
View Forum Message <> Reply to Message

Dave Greenwood wrote:
>
> Out of curiousity - what does gaining "the ObjHeapVar1 space first" do
> for you? How would you use that for something useful?

I would guess so that you can do then

Global_obj = obj_valid(1, /cast)

and get it from anywhere. I wrote a more general hack to access any heap variable this way.
The drawback is, if you ever use IDL_clear or IDL_reset, dangling heap variable will be destroyed.

Pavel