Subject: It seems that there is *a bit* polymorphism in IDL.
Posted by tianyf on Wed, 11 Aug 2004 11:00:30 GMT
View Forum Message <> Reply to Message

I don't know if someone had ever noticed that what will happen if the procedure and the function have the same name. My conclusion is that IDL can figure out which one I am calling - the procedure or the function. This may be a bit useful in some cases. Let's take a look at the following example.

```
;+
; Name:
;  msgbox
;
; Purpose:
;  Another form of DIALOG_MESSAGE.
;
; Type:
;  Function/Procedure
;
; Description:
;  This is a mixed routine which has the same name for function
;  and procedure. When called, it can distinguish which one is
;  called. However, IDL cannot handle functions with the same
;  name but with differenct number or type of parameters.
;
;-
;This is the procedure.
pro msgbox, text, _extra=_extra_
 if n_params() lt 1 then text='Hello, the world!'
 dummy=dialog_message(text,_extra=_extra_)
end
;
;This is the function.
function msgbox, text, _extra=_extra_
 if n_params() lt 1 then text='Hello, the world!'
 dummy=dialog_message(text,_extra=_extra_)
 return,dummy
end
```

Tian.

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by Mark Hadfield on Wed, 11 Aug 2004 21:34:30 GMT
View Forum Message <> Reply to Message

Y.F. Tian wrote:
> I don't know if someone had ever noticed that what will happen if the
> procedure and the function have the same name. My conclusion is that
> IDL can figure out which one I am calling - the procedure or the
> function. This may be a bit useful in some cases.

You are right, it is quite permissible in IDL to have a procedure and a function of the same name. CALL_METHOD is an example of such a pair (the only one I'm aware of in the standard IDL library). Furthermore I think this is a potentially useful feature, though I am sure many, if not most, other IDL programmers will disagree (soon).

However there is a major problem, and it relates to the automatic compilation. Let's say you have your msgbox procedure and function in a file called msgbox.pro. Let's assume that the procedure is first and the function is second (as in your example). If you compile that file manually, you will have both the function and the procedure. OK, so what happens if you put msgbox.pro somewhere on your !PATH so that it will be compiled automatically. Now restart IDL and call the msgbox function. IDL works through the file compiling everything on the way and both the function and the procedure are available. Now restart IDL and call the procedure first. IDL works throught the file until it gets to the procedure. It compiles that & exits *without* compiling the function. As far as I am aware, nothing you can do, short of manually recompiling the file, will persuade IDL to compile the function. You can reverse the order of the procedure and the function, but then you just have the same problem in reverse.

One solution is to have a startup file in which you compile things manually, with a command like

```
resolve_routine, 'msgbox', /COMPILE_FULL_FILE
```

Me, I don't think it's worth the bother.

Note that the above difficulty does not occur with object methods. It is perfectly OK to have two object methods of the same name, one a function and one a procedure. They will both get compiled when <object>__define.pro is compiled. I've been doing this for years.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by David Fanning on Wed, 11 Aug 2004 21:51:12 GMT
View Forum Message <> Reply to Message

Mark Hadfield writes:

> Note that the above difficulty does not occur with object methods. It is
> perfectly OK to have two object methods of the same name, one a function
> and one a procedure. They will both get compiled when
> <object>__define.pro is compiled. I've been doing this for years.

Amen to this! In fact, generalized GetProperty functions
are so easy to write and so useful for obtaining any *single*
object property, that *all* your objects should have one, along
with the usual GetProperty procedure. :-)

http://www.dfanning.com/tips/getproperty.html

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by David Fanning on Thu, 12 Aug 2004 03:05:49 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Amen to this! In fact, generalized GetProperty functions
> are so easy to write and so useful for obtaining any *single*
> object property, that *all* your objects should have one, along
> with the usual GetProperty procedure. :-)
>
> http://www.dfanning.com/tips/getproperty.html

OK, IDL 6.1 arrived this afternoon, so here is a programming
quiz for all of you who are not currently Members in Good
Standing with IEPA (http://www.dfanning.com/misc_tips/iepa.html).
(This includes all of you who have forgotten to pay your
dues this month.)

Use the SCOPE_VARNAME function with the REF_EXTRA keyword
to write a short general purpose GETPROPERTY *procedure* method

that'll return any member variables referenced via
keyword names

  o -> GetPoperty, member_1 = member_1, member_2 = member_2, etc.

where object "o" has fields

      self.member_1
      self.member_2
      etc.

Now you don't need a separate, functional, form of GetProperty!

The usual case of Bavarian Beer for the first correct answer. :-)

Cheers,

David

P.S. Entries will be stamped with a UTC time stamp by the US Naval
Observatory as they arrive, unless a small check accompanies the
entry, etc.

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by David Fanning on Thu, 12 Aug 2004 14:08:21 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Use the SCOPE_VARNAME function with the REF_EXTRA keyword
> to write a short general purpose GETPROPERTY *procedure* method
> that'll return any member variables referenced via
> keyword names
>
>   o -> GetPoperty, member_1 = member_1, member_2 = member_2, etc.
>
> where object "o" has fields
>
>   self.member_1
>   self.member_2
>   etc.
>

> Now you don't need a separate, functional, form of GetProperty!

Well, the contest judges have not exactly been overwhelmed with
the results so far. I expect most participants are trying to
bribe the IT guys to get IDL 6.1 installed before the turn
of the *next* millennium.

Anyway, here is a clue. The GetProperty method's call
prototype should simple be this:

```
PRO MyClass::GetProperty, _REF_EXTRA=extra
```

No other parameters are needed. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by JD Smith on Thu, 12 Aug 2004 17:28:57 GMT
View Forum Message <> Reply to Message

On Wed, 11 Aug 2004 21:05:49 -0600, David Fanning wrote:

> David Fanning writes:
>
>> Amen to this! In fact, generalized GetProperty functions are so easy to
>> write and so useful for obtaining any *single* object property, that
>> *all* your objects should have one, along with the usual GetProperty
>> procedure. :-)
>>
>> http://www.dfanning.com/tips/getproperty.html
>
> OK, IDL 6.1 arrived this afternoon, so here is a programming quiz for all
> of you who are not currently Members in Good Standing with IEPA
> (http://www.dfanning.com/misc_tips/iepa.html). (This includes all of you
> who have forgotten to pay your dues this month.)
>
> Use the SCOPE_VARNAME function with the REF_EXTRA keyword to write a short
> general purpose GETPROPERTY *procedure* method that'll return any member
> variables referenced via keyword names

At long last... legitimized access to the oft-defamed (and poorly named) ROUTINE_NAMES() variable functionality! I can now feel quite justified in my various out-of-scope variable slinging (which is used heavily in IDLWAVE debugging). Very cool new _REF_EXTRA keyword in SCORE_VARFETCH should enable lots of neat functionality, including the one you mention (I presume you meant that one instead of SCOPE_VARNAME).

Also welcome are the left-aligned and zero-padded updates for the FORMAT codes.

On the original topic, I was going to suggest a general purpose execute-based GetProperty function like:

```
function MyClass::GetProperty,_EXTRA=e
  if n_elements(e) eq 0 then return,-1
  prop=(tag_names(e))[0]
  void=execute('self->GetProperty,'+prop+'=ans')
  return,ans
end
```

which uses the GetProperty procedure method (which, in my case, often does more than simply return a field of the class structure). This uses EXECUTE as well, so is to be avoided for VM code. Your requested solution is also trivial:

```
pro MyClass::GetProperty,_REF_EXTRA=e
  tags=tag_names(create_struct(NAME=obj_class(self)))
  for i=0,n_elements(e)-1 do begin
     wh=where(tags eq e[i],cnt)
     if cnt eq 0 then continue
     (scope_varfetch(e[i],/REF_EXTRA))=self.(wh[0])
  endfor
end
```

Note this uses the new (I think, don't have IDL 6.1 yet) functionality for 'CREATE_STRUCT,NAME=' I had just pined for: the ability to create a named structure programmatically without resorting to EXECUTE.

This still doesn't solve the problem of enabling rapid access to object data members: this function will take many hundreds or thousands of times longer to execute than a similar structure field dereference would take. I find that for speed reasons I'm often caching copies of some object's internal fields inside of other objects, which can lead to problems if the cache is not kept up to date. Encapsulation is great, but the penalty for routine calls is too high for some event-driven situations to make good use of it.

I suspect you should also be able to do something clever with
SCOPE_VARFETCH and a GetProperty function, like:

```
function MyClass::GetProperty,_REF_EXTRA=e
  self->GetProperty,_EXTRA=e
  return,scope_varfetch(e[0],/REF_EXTRA)
end
```

Obviously untested, since I don't have 6.1.

Fun stuff.

JD

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by David Fanning on Thu, 12 Aug 2004 17:45:11 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> At long last... legitimized access to the oft-defamed (and poorly
> named) ROUTINE_NAMES() variable functionality!

Oh, geez, JD. Read the rules! I hardly think a Member
in Good Standing Emeritus is qualified to submit an
entry to the contest. (Even if he doesn't have the
software installed.) :-(

> Very cool new _REF_EXTRA keyword in
> SCORE_VARFETCH should enable lots of neat functionality, including the
> one you mention (I presume you meant that one instead of
> SCOPE_VARNAME).

And no fair exposing this little bit of misdirection
to see if anyone was paying attention (or could read
the documentation).

Scheesh! I'm going to go talk to the Chairman. See
if we have any insurance for this kind of thing. :-(

Cheers,

David

P.S. Let's just say the Chairman doesn't like it when he
has to pay out on the very FIRST entry to his contest!

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by JD Smith on Thu, 12 Aug 2004 18:03:04 GMT
View Forum Message <> Reply to Message

On Thu, 12 Aug 2004 11:45:11 -0600, David Fanning wrote:

> JD Smith writes:
>
>> At long last... legitimized access to the oft-defamed (and poorly named)
>> ROUTINE_NAMES() variable functionality!
>
> Oh, geez, JD. Read the rules! I hardly think a Member in Good Standing
> Emeritus is qualified to submit an entry to the contest. (Even if he
> doesn't have the software installed.) :-(

So how do you explain the stack of overdue IEPA membership dues
notices sitting in my tra...errr.... on my desk?

JD

Subject: Re: It seems that there is *a bit* polymorphism in IDL.
Posted by David Fanning on Thu, 12 Aug 2004 19:43:00 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> So how do you explain the stack of overdue IEPA membership dues
> notices sitting in my tra...errr.... on my desk?

Oh, right. I was looking at last year's membership role.
Sorry. I see you got demoted to the Grand Poohba of
the Precious Secrets. Yes, you need to pay up. :-)

Cheers,

David

P.S. Let's just say the Chairman is, uh, not feeling
too well. (How do you spell apoplectic?) I decided to
wait a couple of days to ask about the beer. Hope you

understand.

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
