Subject: call_method
Posted by Pavel A. Romashkin on Thu, 06 Sep 2001 19:37:03 GMT
View Forum Message <> Reply to Message

How does CALL_METHOD work? It would really be nice to be able to use the
same routine as procedure or function, just as Call_method does,
depending on what is happening in it. How did they do it?
Cheers,
Pavel

Subject: Re: call_method
Posted by David Fanning on Fri, 07 Sep 2001 04:09:49 GMT
View Forum Message <> Reply to Message

Mark Hadfield (m.hadfield@niwa.cri.nz) writes:

> All method definitions before myclass__define will be
> compiled so it is OK to have a procedure method and a function method of the
> same name. And I do it all the time.

It must have something to do with all those sheep. :-)

Cheers,

David

P.S. Let's just say that if you wanted to enlighten us
as to *why* you do this all the time, I would be all
ears. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: call_method
Posted by Mark Hadfield on Fri, 07 Sep 2001 04:31:29 GMT
View Forum Message <> Reply to Message

From: "David Fanning" <david@dfanning.com>
> P.S. Let's just say that if you wanted to enlighten us
> as to *why* you do this all the time, I would be all
> ears. :-)

Well, not all the time.

OK, to get specific, I have a general purpose graphics object called MGHgrGraph, based on IDLgrView. MGHgrGraph has a method called NewAtom that creates a graphics atom and attaches it to the graphics tree. Creating an object graphics plot usually consists of creating an MGHgrGraph then calling NewAtom several times. The function form of the NewAtom method returns a reference to the atom; the procedure form doesn't. Most of the time I don't need the reference, so I call the procedure form; sometimes I do need the reference, so I call the function form. (The code is a bit more readable when the procedure form is used, I think.)

The function definition contains all the code to implement NewAtom. The procedure form is a wrapper that looks like this

```
pro MGHgrGraph::NewAtom, P1, P2, P3, P4, RESULT=result, _EXTRA=extra

   compile_opt IDL2

   case n_params() of
      0: result = self->NewAtom( _EXTRA=extra )
      1: result = self->NewAtom( P1, _EXTRA=extra )
      2: result = self->NewAtom( P1, P2, _EXTRA=extra )
      3: result = self->NewAtom( P1, P2, P3, _EXTRA=extra )
      4: result = self->NewAtom( P1, P2, P3, P4, _EXTRA=extra )
   endcase

end
```

The RESULT keyword is for when I change my mind & decide I *do* want the object reference after all.

> It must have something to do with all those sheep. :-)

I don't farm sheep but do I farm olive trees. They have the advantage that they don't move around so much, but on the other hand they aren't as warm & cuddly.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--

Subject: Re: call_method
Posted by Craig Markwardt on Fri, 07 Sep 2001 14:26:32 GMT
View Forum Message <> Reply to Message

"Mark Hadfield" <m.hadfield@niwa.cri.nz> writes:
> OK, to get specific, I have a general purpose graphics object called
> MGHgrGraph, based on IDLgrView. MGHgrGraph has a method called NewAtom that
> creates a graphics atom and attaches it to the graphics tree. Creating an
> object graphics plot usually consists of creating an MGHgrGraph then calling
> NewAtom several times. The function form of the NewAtom method returns a
> reference to the atom; the procedure form doesn't. Most of the time I don't
> need the reference, so I call the procedure form; sometimes I do need the
> reference, so I call the function form. (The code is a bit more readable
> when the procedure form is used, I think.)
>
> The function definition contains all the code to implement NewAtom. The
> procedure form is a wrapper that looks like this
... deleted ...
> The RESULT keyword is for when I change my mind & decide I *do* want the
> object reference after all.

Hi Mark--

Generally speaking when I have some value that I optionally want to
return, I use a keyword. You can even test with arg_present(),
whether the keyword parameter was called with a variable [for example
if it takes a lot of memory.]

I think it is bad form in IDL to intentionally have a procedure and a
function of the same name which do the same thing. As you say, it is
too easy to mix them up...

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: call_method
Posted by Dick Jackson on Fri, 07 Sep 2001 14:38:13 GMT

"Mark Hadfield" <m.hadfield@niwa.cri.nz> wrote in message
news:001501c1374f$113c8ed0$d938a8c0@Hadfield...
> From: "Pavel A. Romashkin" <pavel.romashkin@noaa.gov>
>> David Fanning wrote:
>>
>>> IDL doesn't have any trouble keeping straight the difference
>>> between procedures and functions with the same name.
>>
>> Oh... This never occurred to me. I have never even attempted making a
>> pro and a function with the same name. Duh...
>
> There's no problem in *having* a procedure and a function with the same
> name. The problem is in *compiling* them both. Or to be more exact, in
> having them automatically complied.

Just a reminder of that very handy new feature in IDL 5.4: in those cases
where you can use RESOLVE_ROUTINE, 'myroutine', if you add
/COMPILE_FULL_FILE you should get both the pro and function, no matter which
came first.

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: call_method
Posted by Mark Hadfield on Mon, 10 Sep 2001 03:09:21 GMT

From: "Craig Markwardt" <craigmnet@cow.physics.wisc.edu>
> I think it is bad form in IDL to intentionally have a procedure and a
> function of the same name which do the same thing.

I don't agree. If there weren't any problems in getting both compiled, then
I think it would be a pretty cool thing. Other languages don't have the
distinction between procedures and functions and they don't suffer for it.

But I don't feel all that strongly about it. It's only a computer language
after all.

> As you say, it is
> too easy to mix them up...

Did I say that? If they both did the same thing (except for one returning a value) then it wouldn't matter if you did mix them up.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--