Subject: Re: call_method
Posted by David Fanning on Thu, 06 Sep 2001 20:27:18 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin (pavel.romashkin@noaa.gov) writes:

> 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?

I don't understand what you mean. If you are calling
a procedure method:

```
   Call_Method, theMethod, theObject
```

If you are calling a function method:

```
   result = Call_Method(theMethod, theObject)
```

You can't call a procedure method using the function
syntax, or visa versa.

IDL doesn't have any trouble keeping straight the difference
between procedures and functions with the same name:

```
   PRO junk
   Print, 'In procedure junk'
   END

   FUNCTION junk
   Print, 'In function junk'
   END

   IDL> junk
 In procedure junk
   IDL> a=junk()
 In function junk
```

Clearly, you can write your own routines to work exactly
like Call_Method.

Cheers,

David
--
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 Pavel A. Romashkin on Thu, 06 Sep 2001 20:54:22 GMT
View Forum Message <> Reply to Message

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...
Thanks,
Pavel

Subject: Re: call_method
Posted by David Fanning on Thu, 06 Sep 2001 21:24:56 GMT
View Forum Message <> Reply to Message

Pavel A. Romashkin (pavel.romashkin@noaa.gov) writes:

> Oh... This never occurred to me. I have never even attempted making a
> pro and a function with the same name. Duh...

Ah, well, one day when I didn't feel like working
I got to thinking "what if" and the next thing you
know I had learned something new about IDL. I
filed it under "Reasons You Gotta Love IDL",
hoping someone would ask about it some day. :-)

But it makes sense, because one of the things IDL
saves when a file is compiled is whether it is a
procedure or function. They go in separate tables
internally, I guess.

Cheers,

David

P.S. Let's just say having a childish attitude
about work (or a poor memory) results in more remarkable
discoveries than you might at first imagine. :-)

--
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 03:42:13 GMT
View Forum Message <> Reply to Message

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.

To demonstrate, try the following:

* Save the following code...

```
   pro myroutine
      print, "Hello from MYROUTINE procedure"
   end
   function myroutine
      print, "Hello from MYROUTINE function"
   end
```

...into a file called myroutine.pro and put it somewhere on your path.

* At the IDL prompt type "myroutine". The following appears in the console window:

```
   % Compiled module: MYROUTINE.
   Hello from MYROUTINE procedure
```

* Now type "help, myroutine()". This gives:

```
   % Attempt to call undefined procedure/function: 'MYROUTINE'.
   % Execution halted at: $MAIN$
```

* Reset the IDL session and call the function & procdure in the opposite
order. Same result: the procedure is compiled and the function is not.

* Reverse the order of the procedure and the function in the file, reset the
session and call them again. Now the function will be compiled and the
procedure will not.

* Reset again and compile myroutine.pro manually. Now you get them both

My interpretation: the first time IDL encounters either a function or a
procedure called MYROUTINE, it searches the path for myroutine.pro and then
compiles the file until it encounters EITHER a procedure OR a function
called myroutine. Having compiled that it does not read any more of the
file. Any code that up to & including the first occurrence of myroutine will
be complied; any code after that will not.

So you *can* have a procedure and function of the same name but you'll have
to find some way of ensuring that both are compiled.

Class methods follow the same rules, but normally methods are all gathered
in one file (myclass__define.pro) which is called when the first instance of
this class is created. 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.

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

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG

Subject: Re: call_method
Posted by Kristian Kjaer on Fri, 07 Sep 2001 22:03:23 GMT
View Forum Message <> Reply to Message

Somewhat related, in Ray Sterner's jhuapl library (other places too, I
think), you can do
;
IDL> hismodule_a, /Help
; or
IDL> dummy=hismodule_b(/Help)

;
but you have to know before-hand that hismodule_a is a procedure and
hismodule_b is a function.

Is there any way to make this work without the user having to know
before-hand?

- Kristian Kjær, Risø Natl- Lab., Denmark

David Fanning wrote:
```
>  IDL doesn't have any trouble keeping straight the difference
>  between procedures and functions with the same name:
>
>     PRO junk
>     Print, 'In procedure junk'
>     END
>
>     FUNCTION junk
>     Print, 'In function junk'
>     END
>
>     IDL> junk
>          In procedure junk
>     IDL> a=junk()
>          In function junk
>
>  Clearly, you can write your own routines to work exactly
>  like Call_Method.
>
>  Cheers,
>  David
```