Subject: What subprogram? What parameters and keywords?
Posted by on Fri, 23 Aug 2013 12:18:51 GMT
View Forum Message <> Reply to Message

Let's say IDL has just entered a subroutine or function. Is any of the following information available?

1. The name of the subprogram (like $0 in bash)
2. The parameters as an array or struct or similar (kind of like $*), preferably including keywords.

Subject: Re: What subprogram? What parameters and keywords?
Posted by David Fanning on Fri, 23 Aug 2013 12:52:28 GMT
View Forum Message <> Reply to Message

Mats Löfdahl writes:

> Let's say IDL has just entered a subroutine or function. Is any of the following information available?
>
> 1. The name of the subprogram (like $0 in bash)
> 2. The parameters as an array or struct or similar (kind of like $*), preferably including keywords.

You can find two useful programs in the Coyote Library. WhoAmI returns the name of the program module the WhoAmI program is called in, and WhoCalledMe identifies the name of the program module that called the routine you are now in.

There is no parameter stack that I am aware of, although it wouldn't surprise me if the Scope_*** routines could figure it out.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Fri, 23 Aug 2013 13:57:32 GMT
View Forum Message <> Reply to Message

Den fredagen den 23:e augusti 2013 kl. 14:52:28 UTC+2 skrev David Fanning:
> Mats Löfdahl writes:
>
>> Let's say IDL has just entered a subroutine or function. Is any of the following information available?
>
>> 1. The name of the subprogram (like $0 in bash)
>> 2. The parameters as an array or struct or similar (kind of like $*), preferably including keywords.
>
> You can find two useful programs in the Coyote Library. WhoAmI returns
> the name of the program module the WhoAmI program is called in, and
> WhoCalledMe identifies the name of the program module that called the
> routine you are now in.
>
> There is no parameter stack that I am aware of, although it wouldn't
> surprise me if the Scope_*** routines could figure it out.

Thanks, that solves the first part of the problem! I'll read up on scope_*** and see if I can figure something out for the second part.

Note though that WhoCalledMe and WhoAmI give identical output when I try them. I believe WhoCalledMe needs the following edit:

callingRoutine = (StrSplit(StrCompress(callStack[(index-1) > 0])," ", /Extract))[0]

should be

callingRoutine = (StrSplit(StrCompress(callStack[(index-2) > 0])," ", /Extract))[0]

At least that fixed it for my test case.

Subject: Re: What subprogram? What parameters and keywords?
Posted by Mark Piper on Fri, 23 Aug 2013 14:35:56 GMT
View Forum Message <> Reply to Message

On Friday, August 23, 2013 6:18:51 AM UTC-6, Mats Löfdahl wrote:
>
> 1. The name of the subprogram (like $0 in bash)
> 2. The parameters as an array or struct or similar (kind of like $*), preferably including keywords.

We're looking at introducing the equivalent of argc-argv, where argv[0] is the name of the routine and argv[1-n] are the parameters. Ronn Kling suggested that a list would be a good container for this info. How about new intrinsic routines IDL_ARGC and IDL_ARGV?

mp

Subject: Re: What subprogram? What parameters and keywords?
Posted by Lajos Foldy on Fri, 23 Aug 2013 14:50:26 GMT
View Forum Message <> Reply to Message

On Friday, August 23, 2013 4:35:56 PM UTC+2, Mark Piper wrote:

> We're looking at introducing the equivalent of argc-argv, where argv[0] is the name of the routine and argv[1-n] are the parameters. Ronn Kling suggested that a list would be a good container for this info. How about new intrinsic routines IDL_ARGC and IDL_ARGV?

We have keyword parameters, so a HASH seems to be more appropriate. #0 can be the routine name, #1, #2, ... the positional parameters.

regards,
Lajos

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Fri, 23 Aug 2013 15:09:40 GMT
View Forum Message <> Reply to Message

Den fredagen den 23:e augusti 2013 kl. 15:57:32 UTC+2 skrev Mats Löfdahl:
> Den fredagen den 23:e augusti 2013 kl. 14:52:28 UTC+2 skrev David Fanning:
>> Mats Löfdahl writes:
>
>>> Let's say IDL has just entered a subroutine or function. Is any of the following information available?
>
>>> 1. The name of the subprogram (like $0 in bash)
>>> 2. The parameters as an array or struct or similar (kind of like $*), preferably including keywords.
>
>> You can find two useful programs in the Coyote Library. WhoAmI returns
>> the name of the program module the WhoAmI program is called in, and
>> WhoCalledMe identifies the name of the program module that called the
>> routine you are now in.
>
>> There is no parameter stack that I am aware of, although it wouldn't
>> surprise me if the Scope_*** routines could figure it out.
>
> Thanks, that solves the first part of the problem! I'll read up on scope_*** and see if I can figure something out for the second part.

I didn't give much detail in my original post, but what I'm trying to do is make a subroutine that

writes a log file with as much info as possible about the subprogram it was called from, without having been explicitly told very much. So David's WhoCalledMe is actually better than what I asked for.

And reading up on scope_*** and related routines was good advice, too. Looks like I could do something like this:

* Get the subprogram name from name=WhoCalledMe()
* Get the names of parameters and keywords from routine_info(name,/parameters)
* Get the level of the calling function from scope_level()-1
* Get the current values of the parameters and keywords from scope_varfetch

I haven't tried all the steps yet but it looks promising.

Subject: Re: What subprogram? What parameters and keywords?
Posted by Michael Galloy on Fri, 23 Aug 2013 15:29:45 GMT
View Forum Message <> Reply to Message

On 8/23/13 8:50 am, fawltylanguage@gmail.com wrote:
> On Friday, August 23, 2013 4:35:56 PM UTC+2, Mark Piper wrote:
>
>> We're looking at introducing the equivalent of argc-argv, where
>> argv[0] is the name of the routine and argv[1-n] are the
>> parameters. Ronn Kling suggested that a list would be a good
>> container for this info. How about new intrinsic routines IDL_ARGC
>> and IDL_ARGV?
>
> We have keyword parameters, so a HASH seems to be more appropriate.
> #0 can be the routine name, #1, #2, ... the positional parameters.
>
> regards, Lajos
>

How about a hash for the keywords and a list for the positional parameters?

Would this also include syntax to allow writing a routine that takes an
arbitrary number of parameters?

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: What subprogram? What parameters and keywords?
Posted by David Fanning on Fri, 23 Aug 2013 15:54:41 GMT
View Forum Message <> Reply to Message

Mats Löfdahl writes:

> Note though that WhoCalledMe and WhoAmI give identical output when I try them. I believe WhoCalledMe needs the following edit:
>
> callingRoutine = (StrSplit(StrCompress(callStack[(index-1) > 0])," ", /Extract))[0]
>
> should be
>
> callingRoutine = (StrSplit(StrCompress(callStack[(index-2) > 0])," ", /Extract))[0]
>
> At least that fixed it for my test case.

I noticed this morning that the internal documentation seemed to be the same, but I didn't notice that the code was the same, too. Clearly some kind of cut and paste error. Unfortunately, I didn't have time to check it because my son has me going on bike rides every morning and today was our long one. It's getting harder and harder to keep up with the 20-something kids. :-(

I'll fix this and check it in soon. Thanks!

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: What subprogram? What parameters and keywords?
Posted by Moritz Fischer on Mon, 26 Aug 2013 05:17:13 GMT
View Forum Message <> Reply to Message

Have you tried using the NAMES keyword?

Am 23.08.2013 18:10, schrieb Mats Löfdahl:
>

> The subprograms I'm logging are actually methods. While I'm still
> "in" the method subroutine, I can access the class data in "self".
> With
>
> IDL> help,/obj,self,output=out
>
> I can get the information about it into the variable "out".
>
> I'd like to log this as well, but I don't know how to access this
> info from the logging subroutine that is called by the method
> subroutine.
>
> I thought it would work to do
>
> IDL> help,/obj,self,output=out,level=-1
>
> where level=-1 should go one step up. But I get this:
>
> IDL> print,out SELF UNDEFINED = <Undefined>
>
> Ideas?
>

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Mon, 26 Aug 2013 08:51:57 GMT
View Forum Message <> Reply to Message

Den måndagen den 26:e augusti 2013 kl. 07:17:13 UTC+2 skrev Moritz Fischer:
> Am 23.08.2013 18:10, schrieb Mats Löfdahl:
>
>> The subprograms I'm logging are actually methods. While I'm still
>> "in" the method subroutine, I can access the class data in "self".
>> With
>
>> IDL> help,/obj,self,output=out
>
>> I can get the information about it into the variable "out".
>
>> I'd like to log this as well, but I don't know how to access this
>> info from the logging subroutine that is called by the method
>> subroutine.
>
>> I thought it would work to do
>
>> IDL> help,/obj,self,output=out,level=-1
>
>> where level=-1 should go one step up. But I get this:

>
>> IDL> print,out SELF UNDEFINED = <Undefined>
>
>> Ideas?
>
> Have you tried using the NAMES keyword?

No. Well, I have now. I tried HELP='self' and HELP='*self*' without success.

(I don't really understand why it would solve my problem? From what I understand it is used to filter the information from the help command, not make it find any info that it wouldn't find otherwise. How do you propose I use it?)

Subject: Re: What subprogram? What parameters and keywords?
Posted by Moritz Fischer on Mon, 26 Aug 2013 13:05:32 GMT
View Forum Message <> Reply to Message

Hi again!

help, scope_varfetch('no_such_variable', LEVEL=-1, /ENTER)
<No name> UNDEFINED = <Undefined>

or, to check this on runtime,

defined_flag = n_elements(scope_varfetch('no_such_variable', LEVEL=-1,
/ENTER)) NE 0

cheers

Am 26.08.2013 14:09, schrieb Mats Löfdahl:
> Den fredagen den 23:e augusti 2013 kl. 17:09:40 UTC+2 skrev Mats
> Löfdahl:
>> Den fredagen den 23:e augusti 2013 kl. 15:57:32 UTC+2 skrev Mats
>> Löfdahl:
>>
>>> Den fredagen den 23:e augusti 2013 kl. 14:52:28 UTC+2 skrev David
>>> Fanning:
>>
>>>> Mats Löfdahl writes:
>>
>>>
>>
>>>> > Let's say IDL has just entered a subroutine or function. Is
>>>> > any of the following information available?
>>
>>>

>>
>>>> > 1. The name of the subprogram (like $0 in bash)
>>
>>>> > 2. The parameters as an array or struct or similar (kind of
>>>> > like $*), preferably including keywords.
>>
>>>
>>
>>>> You can find two useful programs in the Coyote Library. WhoAmI
>>>> returns
>>
>>>> the name of the program module the WhoAmI program is called in,
>>>> and
>>
>>>> WhoCalledMe identifies the name of the program module that
>>>> called the
>>
>>>> routine you are now in.
>>
>>>
>>
>>>> There is no parameter stack that I am aware of, although it
>>>> wouldn't
>>
>>>> surprise me if the Scope_*** routines could figure it out.
>>
>>>
>>
>>> Thanks, that solves the first part of the problem! I'll read up
>>> on scope_*** and see if I can figure something out for the second
>>> part.
>>
>>
>>
>> I didn't give much detail in my original post, but what I'm trying
>> to do is make a subroutine that writes a log file with as much info
>> as possible about the subprogram it was called from, without having
>> been explicitly told very much. So David's WhoCalledMe is actually
>> better than what I asked for.
>>
>>
>>
>> And reading up on scope_*** and related routines was good advice,
>> too. Looks like I could do something like this:
>>
>>
>>
>> * Get the subprogram name from name=WhoCalledMe()

>>
>> * Get the names of parameters and keywords from
>> routine_info(name,/parameters)
>>
>> * Get the level of the calling function from scope_level()-1
>>
>> * Get the current values of the parameters and keywords from
>> scope_varfetch
>
> Another problem:
>
> The list of parameters I get from routine_info(name,/parameters)
> includes also those parameters that were not used in the call and
> therefore not defined. When I try to access the values of undefined
> variables, scope_varfetch throws an error.
>
> Is there a way to get the list of used parameters only, or a way to
> avoid the error when calling scope_varfetch with the name of a
> variable that is undefined?
>
>
> Is there a way to
>

Subject: Re: What subprogram? What parameters and keywords?
Posted by Moritz Fischer on Mon, 26 Aug 2013 13:17:46 GMT
View Forum Message <> Reply to Message

now that I tried it: when I compile

```
PRO tmp
  help, scope_varfetch('self',LEVEL=-1)
END
```

and call it from inside a object method, it works fine, i.e. it calls
the overloaded help method of my object.

good luck with your logging,
over and out (as I am on holiday the moment I hit 'send')

Am 26.08.2013 10:51, schrieb Mats Löfdahl:

> No. Well, I have now. I tried HELP='self' and HELP='*self*' without
> success.
>
> (I don't really understand why it would solve my problem? From what I

> understand it is used to filter the information from the help
> command, not make it find any info that it wouldn't find otherwise.
> How do you propose I use it?)
>

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Mon, 26 Aug 2013 13:25:44 GMT
View Forum Message <> Reply to Message

Den måndagen den 26:e augusti 2013 kl. 15:05:32 UTC+2 skrev Moritz Fischer:
>
> or, to check this on runtime,
>
> defined_flag = n_elements(scope_varfetch('no_such_variable', LEVEL=-1,
> /ENTER)) NE 0

Thanks. This works, also without the /ENTER.

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Mon, 26 Aug 2013 13:46:24 GMT
View Forum Message <> Reply to Message

Den måndagen den 26:e augusti 2013 kl. 15:17:46 UTC+2 skrev Moritz Fischer:
> now that I tried it: when I compile
>
> PRO tmp
> help, scope_varfetch('self',LEVEL=-1)
> END
>
> and call it from inside a object method, it works fine, i.e. it calls
> the overloaded help method of my object.

When I try it, it doesn't. But then I don't have any overloaded help method. So maybe that is what I need to fix.

Hmmm. Following the example here:
http://www.exelisvis.com/docs/Creating_Operator_Overlo.html, I put the following in a file red::_overloadHelp.pro (the name of my class is "red"):

```
FUNCTION red::_overloadHelp, Varname

  class = OBJ_CLASS(self)

  text = 'The '+Varname+' variable is an object of class '+class
```

```
  RETURN, text

END
```

The result:

```
IDL> help,scope_varfetch('self',LEVEL=-1)
<No name>       OBJREF    = <ObjHeapVar1(RED)>
```

Did not seem to take any notice of my overloaded help method.

The following also does not work:

```
IDL>  help, self, /obj,  level = -1
SELF            UNDEFINED = <Undefined>
```

So did my overloading not work?

> good luck with your logging,
> over and out (as I am on holiday the moment I hit 'send')

Thanks, have a good holiday.

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Mon, 26 Aug 2013 14:14:05 GMT
View Forum Message <> Reply to Message

Den måndagen den 26:e augusti 2013 kl. 15:46:24 UTC+2 skrev Mats Löfdahl:
>
> So did my overloading not work?

This is new to me. I'm just working with a class written by someone else...

Reading further (http://www.exelisvis.com/docs/IDL_Object.html), I realized I also had to add the line "INHERITS IDL_Object" to the class structure.

Hmmm... And I don't have a "PRO myObjectClass__define", I just have

```
struct = {red, ....., INHERITS IDL_Object}
tmp = obj_new('red')
```

in a function definition. This is in IDL 7.1.1.

I'm a bit out of my depth here...

Subject: Re: What subprogram? What parameters and keywords?
Posted by Michael Galloy on Mon, 26 Aug 2013 17:55:36 GMT
View Forum Message <> Reply to Message

On 8/26/13 8:14 AM, Mats Löfdahl wrote:
> Den måndagen den 26:e augusti 2013 kl. 15:46:24 UTC+2 skrev Mats
> Löfdahl:
>>
>> So did my overloading not work?
>
> This is new to me. I'm just working with a class written by someone
> else...
>
> Reading further (http://www.exelisvis.com/docs/IDL_Object.html), I
> realized I also had to add the line "INHERITS IDL_Object" to the
> class structure.
>
> Hmmm... And I don't have a "PRO myObjectClass__define", I just have
>
> struct = {red, ..., INHERITS IDL_Object} tmp = obj_new('red')
>
> in a function definition. This is in IDL 7.1.1.

Well, technically, if the "struct = { red, ... }" line is executed
before you try to create the object then you have defined your object's
variables, but I would recommend that put the variable definitions in
the appropriately named file: red__define.pro in the "red__define"
procedure. This will make sure IDL (and you) can find the definition
whenever it is needed.

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: What subprogram? What parameters and keywords?
Posted by chris_torrence@NOSPAM on Mon, 26 Aug 2013 22:22:19 GMT
View Forum Message <> Reply to Message

Hi Mats,

None of this is going to work in IDL 7.1.1. Operator overloading was added in IDL 8.0.

But assuming you have access to IDL 8.0 or later, then everything you are doing should work fine.

Cheers,
Chris
ExelisVIS

Subject: Re: What subprogram? What parameters and keywords?
Posted by on Tue, 27 Aug 2013 07:48:43 GMT
View Forum Message <> Reply to Message

On 2013-08-27 00:22, Chris Torrence wrote:
> Hi Mats,
>
> None of this is going to work in IDL 7.1.1. Operator overloading was added in IDL 8.0.
>
> But assuming you have access to IDL 8.0 or later, then everything you are doing should work fine.

Ah. OK, good to know I should stop banging my head against this particular wall.

So, in 7.1, is there any other way to access a class object's "self" data from a subprogram called by a method? I tried sending self as an argument to the subprogram but that didn't help. Of course, I can extract the information I want in the method and then send that to the subprogram. But I'd like to avoid this if possible.

Subject: Re: What subprogram? What parameters and keywords?
Posted by Paul Van Delst[1] on Wed, 28 Aug 2013 20:52:09 GMT
View Forum Message <> Reply to Message

On 08/23/13 11:29, Michael Galloy wrote:
> On 8/23/13 8:50 am, fawltylanguage@gmail.com wrote:
>> On Friday, August 23, 2013 4:35:56 PM UTC+2, Mark Piper wrote:
>>
>>> We're looking at introducing the equivalent of argc-argv, where
>>> argv[0] is the name of the routine and argv[1-n] are the
>>> parameters. Ronn Kling suggested that a list would be a good
>>> container for this info. How about new intrinsic routines IDL_ARGC
>>> and IDL_ARGV?

That would be marvyplate.

>> We have keyword parameters, so a HASH seems to be more appropriate.
>> #0 can be the routine name, #1, #2, ... the positional parameters.
>>
>> regards, Lajos
>>
>
> How about a hash for the keywords and a list for the positional parameters?

+1 for the hash<->keywords idea. A list for keywords doesn't make too
much sense (to me at least).

cheers,

paulv
