Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Jean H. on Fri, 06 Oct 2006 21:11:54 GMT
View Forum Message <> Reply to Message

in "some_proc", if you receive args[] with only 1 element in it. How
would you know if it is the value of foo or of bla?

Jean

Dometz wrote:
> I have already discussed this with a few of you but wanted to see if
> anyone else might know a way of doing the following:
>
> Is it possible to do something like:
>
> foo = "bar"
> bla = 'blu"
> args=[]
> if (N_ELEMENTS(foo) ne 0) then
>   args=[args, foo]
> if (N_ELEMENTS(bla) ne 0) then
>   args=[args, bla]
>
> CALL_PROCEDURE('some_proc', args)
>
> So, the goal is to build dynamically an arguments list that will be
> passed in to CALL_PROCEDURE, CALL_FUNCTION or CALL_METHOD. (I want to
> avoid EXECUTE)
>
> thanks,
>
> dometz
>
> PS: My goal is to write a transparent wrapper for IDL functions. So, I
> would like that when using the wrapper or not, "everything" (including
> error message) stays the same.
>
> PS2: David: Yeah, I still havent given up on this. ;-)
>

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by David Fanning on Fri, 06 Oct 2006 21:34:35 GMT
View Forum Message <> Reply to Message

Dometz writes:

> I have already discussed this with a few of you but wanted to see if

> anyone else might know a way of doing the following:
>
> Is it possible to do something like:
>
> foo = "bar"
> bla = 'blu"
> args=[]
> if (N_ELEMENTS(foo) ne 0) then
> args=[args, foo]
> if (N_ELEMENTS(bla) ne 0) then
> args=[args, bla]
>
> CALL_PROCEDURE('some_proc', args)
>
> So, the goal is to build dynamically an arguments list that will be
> passed in to CALL_PROCEDURE, CALL_FUNCTION or CALL_METHOD. (I want to
> avoid EXECUTE)

Do you mean that your statistical sample of responses
it too low to calculate a meaningful sample error?
Or, do you want to know what happens when they guy
who is going to use your program falls off his bike
on the way to the office and hits his head and passes
parameter two into the function as parameter one in his
confusion?

I have to admit, I am totally confused as to what
you want from us. It's a five line program. We have
covered just about every possibility here, from
normal to malicious users. What is it you
think we have missed?

If you want an dynamical list of arguments, use
a pointer array. But to what purpose? To avoid a
possible error? Forget it. Your crazed bicyclist
will screw you up every time.

Cheers,

David

P.S. Maybe it's just me, but I feel like I'm on the
set of Groundhog Day. :-(

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

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Dominic Metzger on Fri, 06 Oct 2006 21:54:38 GMT
View Forum Message <> Reply to Message

Unfortunately, I do have to care about bicyclist with brain damages.
Since I am going to wrap IDL functions, I need to still return the same
kind of error message as when the IDL function is called directly. I
wish I wouldnt have to deal with invalid code but I have to.
All I am trying to do is to make the wrapper functions as transparent
to the user as possible. I dont care about avoiding errors if the user
enters invalid arguments or not the right number of arguments. I just
want to provide the user with the same error messages than he would
have gotten if he didnt use the wrapper functions.

dometz

P.S. "Groundhog Day"? That was a great movie though!

David Fanning wrote:
> Dometz writes:
>
>> I have already discussed this with a few of you but wanted to see if
>> anyone else might know a way of doing the following:
>>
>> Is it possible to do something like:
>>
>> foo = "bar"
>> bla = 'blu"
>> args=[]
>> if (N_ELEMENTS(foo) ne 0) then
>> args=[args, foo]
>> if (N_ELEMENTS(bla) ne 0) then
>> args=[args, bla]
>>
>> CALL_PROCEDURE('some_proc', args)
>>
>> So, the goal is to build dynamically an arguments list that will be
>> passed in to CALL_PROCEDURE, CALL_FUNCTION or CALL_METHOD. (I want to
>> avoid EXECUTE)
>

> Do you mean that your statistical sample of responses
> it too low to calculate a meaningful sample error?
> Or, do you want to know what happens when they guy
> who is going to use your program falls off his bike
> on the way to the office and hits his head and passes
> parameter two into the function as parameter one in his
> confusion?
>
> I have to admit, I am totally confused as to what
> you want from us. It's a five line program. We have
> covered just about every possibility here, from
> normal to malicious users. What is it you
> think we have missed?
>
> If you want an dynamical list of arguments, use
> a pointer array. But to what purpose? To avoid a
> possible error? Forget it. Your crazed bicyclist
> will screw you up every time.
>
> Cheers,
>
> David
>
> P.S. Maybe it's just me, but I feel like I'm on the
> set of Groundhog Day. :-(
>
>
>
>
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Robbie on Fri, 06 Oct 2006 22:05:57 GMT
View Forum Message <> Reply to Message

I'm a little confused as well. But, I think that the answer is "no".

For example, I've had to limit my XML-RPC server to only call arbitrary functions or procedures with up to 8 positional arguments. This is because I need to generate a line of code with CALL_PROCEDURE for each case of N_PARAMS (or N_ELEMENTS).

I would include the code here, but it is too long. It is available from
the user contributed library or from http://www.barnett.id.au/idl
Have a look at rt_xmlrpcserver.pro

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Carsten Lechte on Fri, 06 Oct 2006 22:07:28 GMT
View Forum Message <> Reply to Message

I have no way to test this right now, but why not:

```
function wrap_this, pos_arg1, pos_arg2, pos_arg3, _ref_extra=e
  return,     this( pos_arg1, pos_arg2, pos_arg3, _strict_extra=e)
end
```

for a function that has three positional parameters?
That should give you the original error messages from this(...)
and you do not need to assume anything about which arguments
are supplied by the caller of wrap_this().

chl

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Robbie on Fri, 06 Oct 2006 22:20:29 GMT
View Forum Message <> Reply to Message

Dear Dometz,

A little tip:
If you've been using IDL for less than 6 months then I wouldn't
recommend going down this line of thought. The only reason you should
be doing this is if you are calling procedures from outside IDL. If
these wrappers are just being used inside IDL then I'd recommend
changing your application design to use OO.

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by David Fanning on Fri, 06 Oct 2006 22:34:46 GMT
View Forum Message <> Reply to Message

Carsten Lechte writes:

> I have no way to test this right now, but why not:
>
> function wrap_this, pos_arg1, pos_arg2, pos_arg3, _ref_extra=e

>   return,   this(pos_arg1, pos_arg2, pos_arg3, _strict_extra=e)
> end
>
> for a function that has three positional parameters?
> That should give you the original error messages from this(...)
> and you do not need to assume anything about which arguments
> are supplied by the caller of wrap_this().

Well, we tried that, but he doesn't like it for some reason.

And, in any case, this sort of begs the question of what
the hell is it you are "wrapping" anyway!? :-)

Cheers,

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

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by David Fanning on Fri, 06 Oct 2006 22:36:47 GMT
View Forum Message <> Reply to Message

Robbie writes:

> A little tip:
> If you've been using IDL for less than 6 months then I wouldn't
> recommend going down this line of thought. The only reason you should
> be doing this is if you are calling procedures from outside IDL. If
> these wrappers are just being used inside IDL then I'd recommend
> changing your application design to use OO.

Or, iTools! Those silent error handlers will give you
something to think about, too. :-)

Cheers,

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

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Dominic Metzger on Fri, 06 Oct 2006 22:54:51 GMT
View Forum Message <> Reply to Message

Basically, all I am doing is:
I automatically parse someones code and replace certain calls (ex:
read_png) with wrapper functions. Inside of these wrapper functions, I
record then timestamp when they were called, with what arguments etc.
Now all this is supposed to happen transparent to the user. Basically,
the user shouldnt even know that this happening or at least it shouldnt
affect him.

dometz

David Fanning wrote:
> Robbie writes:
>
>> A little tip:
>> If you've been using IDL for less than 6 months then I wouldn't
>> recommend going down this line of thought. The only reason you should
>> be doing this is if you are calling procedures from outside IDL. If
>> these wrappers are just being used inside IDL then I'd recommend
>> changing your application design to use OO.
>
> Or, iTools! Those silent error handlers will give you
> something to think about, too. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by David Fanning on Fri, 06 Oct 2006 23:10:16 GMT
View Forum Message <> Reply to Message

Dometz writes:

> Basically, all I am doing is:
> I automatically parse someones code and replace certain calls (ex:
> read_png) with wrapper functions. Inside of these wrapper functions, I
> record then timestamp when they were called, with what arguments etc.

> Now all this is supposed to happen transparent to the user. Basically,
> the user shouldnt even know that this happening or at least it shouldnt
> affect him.

Alright, this is sounding more and more like something that
is being funded by Homeland Security, and I want nothing
more to do with it. I really don't think such a thing is
possible in a weakly typed language in which parameters
can be input or output or both at the same time. Which
"parameter" are you going to store anyway? The one that
went in, or the one that came out? If you are going to
store "all" of them, then the user is going to know about
it, despite your best efforts, because memory usage is
going to balloon so much he won't be able to get any work
done anymore.

I think I would be angling for a new project.

Cheers,

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

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Dominic Metzger on Fri, 06 Oct 2006 23:19:09 GMT
View Forum Message <> Reply to Message

Nothing to do with Homeland Security. The user has to turn on this
feature and it anyways is impossible to make it completely invisible to
the user since error messages will say something like:

Error at some_wrapper line 16.

So there is no chance of making it completely transparent.
I wont store the (ingoing) parameters I will just print some of them to
file such as the filename that was opened.

best regards,

dometz

David Fanning wrote:

> Dometz writes:
>
>> Basically, all I am doing is:
>> I automatically parse someones code and replace certain calls (ex:
>> read_png) with wrapper functions. Inside of these wrapper functions, I
>> record then timestamp when they were called, with what arguments etc.
>> Now all this is supposed to happen transparent to the user. Basically,
>> the user shouldnt even know that this happening or at least it shouldnt
>> affect him.
>
> Alright, this is sounding more and more like something that
> is being funded by Homeland Security, and I want nothing
> more to do with it. I really don't think such a thing is
> possible in a weakly typed language in which parameters
> can be input or output or both at the same time. Which
> "parameter" are you going to store anyway? The one that
> went in, or the one that came out? If you are going to
> store "all" of them, then the user is going to know about
> it, despite your best efforts, because memory usage is
> going to balloon so much he won't be able to get any work
> done anymore.
>
> I think I would be angling for a new project.
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by David Fanning on Fri, 06 Oct 2006 23:33:41 GMT
View Forum Message <> Reply to Message

Dometz writes:

> Nothing to do with Homeland Security. The user has to turn on this
> feature and it anyways is impossible to make it completely invisible to
> the user since error messages will say something like:
>
> Error at some_wrapper line 16.
>
> So there is no chance of making it completely transparent.
> I wont store the (ingoing) parameters I will just print some of them to

> file such as the filename that was opened.

Well, have at it, then. Let us know how it goes. :-)

Cheers,

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

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Dominic Metzger on Fri, 06 Oct 2006 23:36:01 GMT
View Forum Message <> Reply to Message

Well, I guess I will try my best. I will let you know if I find a way.

Thanks for your help!

dometz

David Fanning wrote:
> Dometz writes:
>
>> Nothing to do with Homeland Security. The user has to turn on this
>> feature and it anyways is impossible to make it completely invisible to
>> the user since error messages will say something like:
>>
>> Error at some_wrapper line 16.
>>
>> So there is no chance of making it completely transparent.
>> I wont store the (ingoing) parameters I will just print some of them to
>> file such as the filename that was opened.
>
> Well, have at it, then. Let us know how it goes. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by news.qwest.net on Sat, 07 Oct 2006 21:41:39 GMT
View Forum Message <> Reply to Message

"Dometz" <dometz@gmail.com> wrote in message
news:1160175290.967818.184890@m73g2000cwd.googlegroups.com.. .
> Basically, all I am doing is:
> I automatically parse someones code and replace certain calls (ex:
> read_png) with wrapper functions. Inside of these wrapper functions, I
> record then timestamp when they were called, with what arguments etc.
> Now all this is supposed to happen transparent to the user. Basically,
> the user shouldnt even know that this happening or at least it shouldnt
> affect him.
>
> dometz

Well, the user is going to know, because the user will have to call the
wrapper.

The following is a horrible suggestion, I forbid you to do it. :)

but, why not just modify the actual function (if it is indeed read_png.pro
or any of the other functions with pro files). They are in the lib folder
of the IDL installation.
Just make sure that you take care when upgrading IDL versions.

I would make a CREATE_LOG_ENTRY procedure that does everything you want,
and paste it into each of the functions you wanted to wrap.

Inside of read_pict.pro
"PRO READ_PICT, filename, resultimage, r, g, b, DEBUG = DEBUG"

add a line like:
CREATE_LOG_ENTRY, 'Read_pict', filename, resultimage, r, g, b, debug

where your routine is something like

pro CREATE_LOG_ENTRY,
routinename,generic1,generic2,generic3,generic4,generic5,gen eric6,generic7,generic8
openw, lun,'logfile'
printf,lun, routinename
printf,lun, n_elements(generic1)
printf,lun, n_elements(generic2)
printf,lun, n_elements(generic3)
etc.

Cheers,
bob

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by Dominic Metzger on Sun, 08 Oct 2006 17:17:34 GMT
View Forum Message <> Reply to Message

Hey Bob,

Thanks for the suggestion... I guess that would work... hmmm... and it would also solve the problem of transparency... it is something to consider.

> Well, the user is going to know, because the user will have to call the
> wrapper.

Yes, the wrappers have to be in the users code, but I do this automagically before the users code is run. In other words, I create a copy of the users code with the wrapper functions in it.

thanks,

dometz

R.G. Stockwell wrote:
> "Dometz" <dometz@gmail.com> wrote in message
> news:1160175290.967818.184890@m73g2000cwd.googlegroups.com.. .
>> Basically, all I am doing is:
>> I automatically parse someones code and replace certain calls (ex:
>> read_png) with wrapper functions. Inside of these wrapper functions, I
>> record then timestamp when they were called, with what arguments etc.
>> Now all this is supposed to happen transparent to the user. Basically,
>> the user shouldnt even know that this happening or at least it shouldnt
>> affect him.
>>
>> dometz
>
>
> Well, the user is going to know, because the user will have to call the
> wrapper.
>
> The following is a horrible suggestion, I forbid you to do it. :)
>
> but, why not just modify the actual function (if it is indeed read_png.pro
> or any of the other functions with pro files). They are in the lib folder

> of the IDL installation.
> Just make sure that you take care when upgrading IDL versions.
>
> I would make a CREATE_LOG_ENTRY procedure that does everything you want,
> and paste it into each of the functions you wanted to wrap.
>
> Inside of read_pict.pro
> "PRO READ_PICT, filename, resultimage, r, g, b, DEBUG = DEBUG"
>
> add a line like:
> CREATE_LOG_ENTRY, 'Read_pict', filename, resultimage, r, g, b, debug
>
>
> where your routine is something like
>
> pro CREATE_LOG_ENTRY,
> routinename,generic1,generic2,generic3,generic4,generic5,gen eric6,generic7,generic8
> openw, lun,'logfile'
> printf,lun, routinename
> printf,lun, n_elements(generic1)
> printf,lun, n_elements(generic2)
> printf,lun, n_elements(generic3)
> etc.
>
>
>
> Cheers,
> bob

Subject: Re: call_procedure with a dynamically created arguments list?
Posted by JD Smith on Mon, 09 Oct 2006 18:37:17 GMT
View Forum Message <> Reply to Message

On Fri, 06 Oct 2006 14:54:38 -0700, Dometz wrote:

> Unfortunately, I do have to care about bicyclist with brain damages.
> Since I am going to wrap IDL functions, I need to still return the same
> kind of error message as when the IDL function is called directly. I
> wish I wouldnt have to deal with invalid code but I have to.
> All I am trying to do is to make the wrapper functions as transparent
> to the user as possible. I dont care about avoiding errors if the user
> enters invalid arguments or not the right number of arguments. I just
> want to provide the user with the same error messages than he would
> have gotten if he didnt use the wrapper functions.

Languages like lisp allow you to "advise" functions to add this type of
functionality. I think Dometz wants a single magic super-duper

routine, which can take any number of arguments/keyword parameters, log them, then pass them transparently on to the "real" routine. This is a tall order. You can't concatenate all arguments into an array: what if one is a float, and one is a string? If you really, really, really want to go this route, you might try something like the one below (and a similar wrap_function), limited to 25 arguments maximum.

JD

```
pro wrap_procedure,routine,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10, $
                v11,v12,v13,v14,v15,v16,v17,v18,v19,v20, $
                v21,v22,v23,v24,v25,_REF_EXTRA=e
  on_error,2
  log_all,routine, v1,v2,v3,v4,v5,v6,v7,v8,v9,v10, $
        v11,v12,v13,v14,v15,v16,v17,v18,v19,v20, $
        v21,v22,v23,v24,v25,e
  narg=n_params()-1
  case narg of
    0: call_procedure,routine,_STRICT_EXTRA=e
    1: call_procedure,routine,v1,_STRICT_EXTRA=e
    2: call_procedure,routine,v1,v2,_STRICT_EXTRA=e
    3: call_procedure,routine,v1,v2,v3, $
                  _STRICT_EXTRA=e
    4: call_procedure,routine,v1,v2,v3,v4, $
                  _STRICT_EXTRA=e
    5: call_procedure,routine,v1,v2,v3,v4,v5, $
                  _STRICT_EXTRA=e
    6: call_procedure,routine,v1,v2,v3,v4,v5,v6, $
                  _STRICT_EXTRA=e
    7: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  _STRICT_EXTRA=e
    8: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                 v8,_STRICT_EXTRA=e
    9: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                 v8,v9,_STRICT_EXTRA=e
    10: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,_STRICT_EXTRA=e
    11: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,v11,_STRICT_EXTRA=e
    12: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,v11,v12,_STRICT_EXTRA=e
    13: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,v11,v12,v13,_STRICT_EXTRA=e
    14: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,v11,v12,v13,v14,_STRICT_EXTRA=e
    15: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                  v8,v9,v10,v11,v12,v13,v14,v15,_STRICT_EXTRA=e
    16: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
```

```
                v8,v9,v10,v11,v12,v13,v14,v15,v16,_STRICT_EXTRA=e
    17: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,_STRICT_EXTRA=e
    18: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18, $
                _STRICT_EXTRA=e
    19: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19, $
                _STRICT_EXTRA=e
    20: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20, $
                _STRICT_EXTRA=e
    21: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21, $
                _STRICT_EXTRA=e
    22: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21, $
                v22,_STRICT_EXTRA=e
    23: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21, $
                v22,v23,_STRICT_EXTRA=e
    24: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21, $
                v22,v23,v24,_STRICT_EXTRA=e
    25: call_procedure,routine,v1,v2,v3,v4,v5,v6,v7, $
                v8,v9,v10,v11,v12,v13,v14,v15,v16,v17,v18,v19,v20,v21, $
                v22,v23,v24,v25,_STRICT_EXTRA=e
  endcase
end
```