
Subject: eval

Posted by [Dominic Metzger](#) on Wed, 04 Oct 2006 00:53:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

Is there something like the Unix eval command:

"The eval utility will construct a command by concatenating arguments together, separating each with a space character. The constructed command will be read and executed by the shell."

in IDL? So that I basically can create a command dynamically by creating a string that will then be "evaluated"?

Something like:

```
IDL> foo="print, bar"
```

```
IDL> eval(foo)
```

```
bar
```

thanks,

dometz

Subject: Re: eval

Posted by [David Fanning](#) on Wed, 04 Oct 2006 01:13:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

Dometz writes:

```
> Is there something like the Unix eval command:
> "The eval utility will construct a command by concatenating arguments
> together, separating each with a space character. The constructed
> command will be read and executed by the shell."
> in IDL? So that I basically can create a command dynamically by
> creating a string that will then be "evaluated"?
>
> Something like:
> IDL> foo="print, bar"
> IDL> eval(foo)
> bar
```

See EXECUTE.

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: eval
Posted by [Dominic Metzger](#) on Wed, 04 Oct 2006 01:25:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thanks David...

Would this also be possible with `call_function` or `call_procedure` since `EXECUTE` causes certain limitations and inefficiencies.

Something like:

```
IDL> foo="bar, arg1, arg2, arg3, arg4=foo"
IDL> call_procedure(foo)
```

I want to dynamically build the list of arguments as a string and the call the whole thing. Is this possible with `call_function` or `call_procedure`?

Or is it possible to pass in an array containing the arguments (including named args) somehow?

thanks,

dometz

PS: Is it even possible to get the return value when using `execute` for a function?

David Fanning wrote:

> Dometz writes:

>

>> Is there something like the Unix `eval` command:

>> "The `eval` utility will construct a command by concatenating arguments together, separating each with a space character. The constructed command will be read and executed by the shell."

>> in IDL? So that I basically can create a command dynamically by creating a string that will then be "evaluated"?

>>

>> Something like:

>> IDL> foo="print, bar"

>> IDL> eval(foo)

>> bar

>

> See `EXECUTE`.

>
> 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: eval

Posted by [David Fanning](#) on Wed, 04 Oct 2006 01:51:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

Dometz writes:

> Would this also be possible with call_function or call_procedure since
> EXECUTE causes certain limitations and inefficiencies.
>
> Something like:
> IDL> foo="bar, arg1, arg2, arg3, arg4=foo"
> IDL> call_procedure(foo)
>
> I want to dynamically build the list of arguments as a string and the
> call the whole thing. Is this possible with call_function or
> call_procedure?
> Or is it possible to pass in an array containing the arguments
> (including named args) somehow?

Call_Procedure, Call_Method, and Call_Function, along with Execute, I call "virtual" functions. With Execute you create the actual IDL command you want to execute as a string. With the other three virtual functions the name of the procedure/function/method is passed as a string (usually so it can be determined at run-time) but the rest of the parameters are passed normally, as if to the correct procedure/function/method.

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: eval

Posted by [Dominic Metzger](#) on Wed, 04 Oct 2006 04:29:38 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi David,

"but the rest of the parameters are passed normally,
> as if to the correct procedure/function/method."

1) Is there a hack around this so that I could use call_function,
call_procedure or call_method dynamically?

2) Or at least build an array of the arguments so that I could do that
alternatively? Like in Perl where you can pass in an array of
arguments... hmmm, that might not work for name arguments though...

thanks,

dometz

PS: If I call a function with EXECUTE, is there a way to get the return
value?

David Fanning wrote:

> Dometz writes:

>

>> Would this also be possible with call_function or call_procedure since

>> EXECUTE causes certain limitations and inefficiencies.

>>

>> Something like:

>> IDL> foo="bar, arg1, arg2, arg3, arg4=foo"

>> IDL> call_procedure(foo)

>>

>> I want to dynamically build the list of arguments as a string and the

>> call the whole thing. Is this possible with call_function or

>> call_procedure?

>> Or is it possible to pass in an array containing the arguments

>> (including named args) somehow?

>

> Call_Procedure, Call_Method, and Call_Function, along

> with Execute, I call "virtual" functions. With Execute

> you create the actual IDL command you want to execute

> as a string. With the other three virtual functions

> the name of the procedure/function/method is passed

> as a string (usually so it can be determined at run-time)

> but the rest of the parameters are passed normally,

> as if to the correct procedure/function/method.

>

> 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: eval
Posted by [Maarten\[1\]](#) on Wed, 04 Oct 2006 07:53:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dometz wrote:

> PS: If I call a function with EXECUTE, is there a way to get the return
> value?

execute, well, executes a full statement. If you set the string to
"retVal = function(...)", then after the call the local variable retVal
will hold the result of the function call.

Maarten

(PS, there are many good reasons to try to avoid execute, call_xxx
offer more flexibility, and far more speed. Perhaps if you give more
details of what you want to achieve, and why it is hard to use the
call_xxx variants, we can offer suggestions.)

Subject: Re: eval
Posted by [Braedley](#) on Wed, 04 Oct 2006 19:12:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

Maarten wrote:

> Dometz wrote:
>> PS: If I call a function with EXECUTE, is there a way to get the return
>> value?
>
> execute, well, executes a full statement. If you set the string to
> "retVal = function(...)", then after the call the local variable retVal
> will hold the result of the function call.
>
> Maarten
>
> (PS, there are many good reasons to try to avoid execute, call_xxx
> offer more flexibility, and far more speed. Perhaps if you give more

> details of what you want to achieve, and why it is hard to use the
> call_xxx variants, we can offer suggestions.)

Sometimes it's not possible to avoid using execute, especially for event driven programs (and I have accepted this for one of my widget programs), but often you can use call_xxx to do most of what you want execute to do. Also, I brought this up a little earlier here.

http://groups.google.com/group/comp.lang.idl-pvwave/browse_thread/thread/16508f93c875a044?vc=2

Cheers
Braedley

Subject: Re: eval
Posted by [Dominic Metzger](#) on Wed, 04 Oct 2006 19:22:19 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thank you all for your help, so here is what I am trying to do

I want to write some wrapper functions for some IDL procedure / functions. I just saw _EXTRA which I might be able to use to do what I need to do .?

So for example for OPENU, I want to create a wrapper function that will do some logging and then call the original OPENU:

example:

```
PRO LOG_OPENU, UNIT, FILE, APPEND=APPEND, COMPRESS=COMPRESS,  
BUFSIZE=BUFSIZE, DELETE=DELETE, ERROR=ERROR,  
F77_UNFORMATTED=F77_UNFORMATTED, GET_LUN=GET_LUN, MORE=MORE,  
NOEXPAND_PATH=NOEXPAND_PATH, STDIO=STDIO, SWAP_ENDIAN=SWAP_ENDIAN, $  
    SWAP_IF_BIG_ENDIAN=SWAP_IF_BIG_ENDIAN,  
SWAP_IF_LITTLE_ENDIAN=SWAP_IF_LITTLE_ENDIAN, VAX_FLOAT=VAX_FLOAT,  
WIDTH=WIDTH, XDR=XDR, RAWIO=RAWIO  
    print, FILE  
    print, SWAP_IF_BIG_ENDIAN  
    ;print etc...  
    ;Now here would like to call the original method but I have to call  
it only with the named arguments that are set  
    OPENU, ...?
```

So, either I could have a LOOOONG list of if or case statements or I maybe I could use _EXTRA?

Or:

I would need to build the method call dynamically:

1) Is there a hack around this so that I could use call_function,
call_procedure or call_method dynamically?

2) Or at least build an array of the arguments so that I could do that alternatively? Like in Perl where you can pass in an array of arguments... hmmm, that might not work for name arguments though...

3) Or a way to use `_EXTRA`

thanks,

dometz

Maarten wrote:

> Dometz wrote:

>> PS: If I call a function with EXECUTE, is there a way to get the return
>> value?

>

> execute, well, executes a full statement. If you set the string to
> "retVal = function(...)", then after the call the local variable retVal
> will hold the result of the function call.

>

> Maarten

>

> (PS, there are many good reasons to try to avoid execute, call_xxx
> offer more flexibility, and far more speed. Perhaps if you give more
> details of what you want to achieve, and why it is hard to use the
> call_xxx variants, we can offer suggestions.)

Subject: Re: eval

Posted by [David Fanning](#) on Wed, 04 Oct 2006 21:21:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

Dometz writes:

> So for example for OPENU, I want to create a wrapper function that will
> do some logging and then call the original OPENU:

>

> example:

> PRO LOG_OPENU, UNIT, FILE, APPEND=APPEND, COMPRESS=COMPRESS,
> BUFSIZE=BUFSIZE, DELETE=DELETE, ERROR=ERROR,
> F77_UNFORMATTED=F77_UNFORMATTED, GET_LUN=GET_LUN, MORE=MORE,
> NOEXPAND_PATH=NOEXPAND_PATH, STDIO=STDIO, SWAP_ENDIAN=SWAP_ENDIAN,
> \$
> SWAP_IF_BIG_ENDIAN=SWAP_IF_BIG_ENDIAN,
> SWAP_IF_LITTLE_ENDIAN=SWAP_IF_LITTLE_ENDIAN, VAX_FLOAT=VAX_FLOAT,
> WIDTH=WIDTH, XDR=XDR, RAWIO=RAWIO
> print, FILE

> print, SWAP_IF_BIG_ENDIAN
> ;print etc...
> ;Now here would like to call the original method but I have to call
> it only with the named arguments that are set
> OPENU, ...?
>
> So, either I could have a LOOOONG list of if or case statements or I
> maybe I could use _EXTRA?

This is exactly what _EXTRA is used for. Note that if you want to use output keywords and get information *back* from your LOG_OPENU procedure, you should use _REF_EXTRA instead of _EXTRA. And if you want to be sure that ONLY keywords for OPENU are passed in, you should use _STRICT_EXTRA. I just noticed there is no _STRICT_REF_EXTRA, but I guess I've never really needed it, so probably that's why. :-)

> Or:
> I would need to build the method call dynamically:
> 1) Is there a hack around this so that I could use call_function,
> call_procedure or call_method dynamically?

There is no reason to use CALL_PROCEDURE here, since you know *exactly* what procedure you want to call: OPENU. You might use CALL_PROCEDURE if you didn't know until run-time whether you wanted to call OPENU, or OPENR, or OPENW, for example.

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: eval

Posted by [Michael Galloy](#) on Wed, 04 Oct 2006 22:56:59 GMT

[View Forum Message](#) <> [Reply to Message](#)

David Fanning wrote:

> This is exactly what _EXTRA is used for. Note that if you
> want to use output keywords and get information *back* from
> your LOG_OPENU procedure, you should use _REF_EXTRA instead
> of _EXTRA. And if you want to be sure that ONLY keywords
> for OPENU are passed in, you should use _STRICT_EXTRA. I just
> noticed there is no _STRICT_REF_EXTRA, but I guess I've never

> really needed it, so probably that's why. :-)

You can combine `_REF_EXTRA` and `_STRICT_EXTRA` because the `_REF_EXTRA` goes in the routine header line and the `_STRICT_EXTRA` goes in the function call. Like:

```
pro some_routine_wrapper, _ref_extra=e
  compile_opt strictarr
```

```
  some_routine, _strict_extra=e
end
```

Mike

--

www.michaelgalloy.com

Subject: Re: eval

Posted by [David Fanning](#) on Wed, 04 Oct 2006 23:13:04 GMT

[View Forum Message](#) <> [Reply to Message](#)

mgalloy@gmail.com writes:

> You can combine `_REF_EXTRA` and `_STRICT_EXTRA` because the `_REF_EXTRA`
> goes in the routine header line and the `_STRICT_EXTRA` goes in the
> function call. Like:

>

```
> pro some_routine_wrapper, _ref_extra=e
>   compile_opt strictarr
```

>

```
>   some_routine, _strict_extra=e
```

```
> end
```

Oh, right. I never get this right when I try to use it.
I should write it down. :-)

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: eval

Posted by wlandsman@jhu.edu on Thu, 05 Oct 2006 02:40:05 GMT

[View Forum Message](#) <> [Reply to Message](#)

"Dometz" <dometz@gmail.com> wrote in message

news:1159989739.132283.84940@i3g2000cwc.googlegroups.com...

>

> So for example for OPENU, I want to create a wrapper function that will

> do some logging and then call the original OPENU:

>

> example:

> PRO LOG_OPENU, UNIT, FILE, APPEND=APPEND, COMPRESS=COMPRESS,

> BUFSIZE=BUFSIZE, DELETE=DELETE, ERROR=ERROR,

> F77_UNFORMATTED=F77_UNFORMATTED, GET_LUN=GET_LUN, MORE=MORE,

> NOEXPAND_PATH=NOEXPAND_PATH, STDIO=STDIO, SWAP_ENDIAN=SWAP_ENDIAN,

\$

> SWAP_IF_BIG_ENDIAN=SWAP_IF_BIG_ENDIAN,

> SWAP_IF_LITTLE_ENDIAN=SWAP_IF_LITTLE_ENDIAN, VAX_FLOAT=VAX_FLOAT,

> WIDTH=WIDTH, XDR=XDR, RAWIO=RAWIO

> print, FILE

> print, SWAP_IF_BIG_ENDIAN

> ;print etc...

> ;Now here would like to call the original method but I have to call

> it only with the named arguments that are set

> OPENU, ...?

>

> So, either I could have a LOOOONG list of if or case statements or I

> maybe I could use _EXTRA?

_EXTRA is the way to go, but as a side note, you wouldn't have needed a long list of case statements but just a single call to OPENU with all the keywords present. OPENU will consider the keywords set only if they are present *and* have a non-zero value. So if they weren't defined in the call to the wrapper procedure, they won't be considered set by ENU. --Wayne

Subject: Re: eval

Posted by [Dominic Metzger](#) on Thu, 05 Oct 2006 05:36:40 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thank you for all your help.

_EXTRA definitely seems the way to go! :-)

I tried:

```
PRO LOG_OPENU, UNIT, FILE, _REF_EXTRA = ex
```

```
PRINT, 'OPENU'
```

```
OPENU, UNIT, FILE, _EXTRA=ex
```

END

and it seems to work. I guess I will try the strict version of `_EXTRA` next.

Wayne Landsman: This might work in some cases but definitely not for all.

Example wrapping spawn:

from the specs of spawn:

"If UNIT is present, Command must be present, and neither Result or ErrResult are allowed."

http://idlastro.gsfc.nasa.gov/idl_html_help/SPAWN.html

In this case, one would have to check if UNIT was present or not... thereby, a case statement is needed. Now, there a quite of few cases like this. Therefore, the better approach seems to be the use of `_EXTRA`.

best regards,

dometz

Wayne Landsman wrote:

> "Dometz" <dometz@gmail.com> wrote in message

> news:1159989739.132283.84940@i3g2000cwc.googlegroups.com...

>>

>> So for example for OPENU, I want to create a wrapper function that will

>> do some logging and the call the original OPENU:

>>

>> example:

>> PRO LOG_OPENU, UNIT, FILE, APPEND=APPEND, COMPRESS=COMPRESS,

>> BUFSIZE=BUFSIZE, DELETE=DELETE, ERROR=ERROR,

>> F77_UNFORMATTED=F77_UNFORMATTED, GET_LUN=GET_LUN, MORE=MORE,

>> NOEXPAND_PATH=NOEXPAND_PATH, STDIO=STDIO, SWAP_ENDIAN=SWAP_ENDIAN,
\$

>> SWAP_IF_BIG_ENDIAN=SWAP_IF_BIG_ENDIAN,

>> SWAP_IF_LITTLE_ENDIAN=SWAP_IF_LITTLE_ENDIAN, VAX_FLOAT=VAX_FLOAT,

>> WIDTH=WIDTH, XDR=XDR, RAWIO=RAWIO

>> print, FILE

>> print, SWAP_IF_BIG_ENDIAN

>> ;print etc...

>> ;Now here would like to call the original method but I have to call

>> it only with the named arguments that are set

>> OPENU, ...?

>>
>> So, either I could have a LOOOONG list of if or case statements or I
>> maybe I could use _EXTRA?
>
> _EXTRA is the way to go, but as a side note, you wouldn't have needed a long
> list of case statements but just a single call to OPENU with all the
> keywords present. OPENU will consider the keywords set only if they are
> present *and* have a non-zero value. So if they weren't defined in the
> call to the wrapper procedure, they won't be considered set by
> ENU. --Wayne

Subject: Re: eval

Posted by [JD Smith](#) on Mon, 09 Oct 2006 17:51:26 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Wed, 04 Oct 2006 15:21:48 -0600, David Fanning wrote:

> Dometz writes:
>
>> [quoted text muted]
>
> This is exactly what _EXTRA is used for. Note that if you
> want to use output keywords and get information *back* from
> your LOG_OPENU procedure, you should use _REF_EXTRA instead
> of _EXTRA. And if you want to be sure that ONLY keywords
> for OPENU are passed in, you should use _STRICT_EXTRA. I just
> noticed there is no _STRICT_REF_EXTRA, but I guess I've never
> really needed it, so probably that's why. :-)

_REF_EXTRA is only used in routine definitions, since it defines how
inherited keywords will be passed in (by value or by reference).

_STRICT_EXTRA is only used in routine calls, since it determines
whether the passed keyword names will be checked against the list of
accepted keywords in the called function. So _STRICT_REF_EXTRA would have
no purpose.

JD
