Subject: Function call string
Posted by Gordon Farquharson on Thu, 09 Apr 2015 20:20:09 GMT
View Forum Message <> Reply to Message

Is there a way to access the string that was used to call a function or procedure in IDL? For example, lets say I call:

display, 'A carrot is orange'

Is there a way to access the string "display, 'A carrot is orange'" from within the display procedure?

The reason I'd like to do this is that I'd like to record how a function or procedure was called in the meta data that the function or procedure generates, so that it is easy to make the same call again in the future.

(I seem to recall that there was a discussion on this topic previously, but I can't find the thread.)

I realize that I could construct the function call string from the arguments, but the output is not generic or pretty because one has to decide on things like how many decimal places to use to represent floating point values, and one has to take care of each keyword setting.

Maybe a better self-documenting way to do this is to have a configuration file, e.g., an XML file that the procedure or function reads, and that accompanies the output. While I have done this previously for other routines, it seems a little overkill for my current application. Maybe it is the best solution though.

(Not sure if David is still posting tips for his web page, but this would be a cool one to add if there is a neat solution. Maybe he already has it there.)

Gordon

Subject: Re: Function call string
Posted by penteado on Fri, 10 Apr 2015 05:25:46 GMT
View Forum Message <> Reply to Message

Hello,

One way, not necessarily a good way, is to use the help output:

```
pro functioncallstring,arg1,arg2,key1=key1,key2=key2
help,/recall,output=o
print,o[1]
end
```

So that when calling it the result would be

IDL> functioncallstring,10*!dpi,cos([1d0,-1d0]),key1='some '+'string'
1 functioncallstring,10*!dpi,cos([1d0,-1d0]),key1='some '+'string'

I can see some caveats:

1) The usual warning about using help's output:
"Note: The OUTPUT keyword is primarily for use in capturing HELP output in order to display it someplace else, such as in a text widget. This keyword is not intended to be used in obtaining programmatic information about the IDL session, and is formatted to be human readable. The format and content of this text is not guaranteed to remain static from release to release and may change at any time, without warning."
(file:///usr/local/exelis/idl84/help/online_help/IDL/idl.htm
#Reference%20Material/H/HELP.htm#H_835179117_832171)

2) This would only show how this procedure was called from the command line, because it is just showing the last command typed on the console. If you are looking to how the procedure was called from some other procedure, this will fail.

Subject: Re: Function call string
Posted by penteado on Fri, 10 Apr 2015 05:30:19 GMT
View Forum Message <> Reply to Message

Hello,

One way, not necessarily a good way, is to use the help output:

```
pro functioncallstring,arg1,arg2,key1=key1,key2=key2
help,/recall,output=o
print,o[1]
end
```

So that when calling it the result would be

```
IDL> functioncallstring,10*!dpi,cos([1d0,-1d0]),key1='some '+'string'
1       functioncallstring,10*!dpi,cos([1d0,-1d0]),key1='some '+'string'
```

I can see some caveats:

1) The usual warning about using help's output:
"Note: The OUTPUT keyword is primarily for use in capturing HELP output in order to display it someplace else, such as in a text widget. This keyword is not intended to be used in obtaining programmatic information about the IDL session, and is formatted to be human readable. The format and content of this text is not guaranteed to remain static from release to release and may change at any time, without warning."
(http://www.exelisvis.com/docs/HELP.html#H_835179117_832171)

2) This would only show how this procedure was called from the command line, because it is just

showing the last command typed on the console. If you are looking to how the procedure was called from some other procedure, this will fail.

On Thursday, April 9, 2015 at 5:20:11 PM UTC-3, Gordon Farquharson wrote:
> Is there a way to access the string that was used to call a function or procedure in IDL? For example, lets say I call:
>
> display, 'A carrot is orange'
>
> Is there a way to access the string "display, 'A carrot is orange'" from within the display procedure?
>
> The reason I'd like to do this is that I'd like to record how a function or procedure was called in the meta data that the function or procedure generates, so that it is easy to make the same call again in the future.
>
> (I seem to recall that there was a discussion on this topic previously, but I can't find the thread.)
>
> I realize that I could construct the function call string from the arguments, but the output is not generic or pretty because one has to decide on things like how many decimal places to use to represent floating point values, and one has to take care of each keyword setting.
>
> Maybe a better self-documenting way to do this is to have a configuration file, e.g., an XML file that the procedure or function reads, and that accompanies the output. While I have done this previously for other routines, it seems a little overkill for my current application. Maybe it is the best solution though.
>
> (Not sure if David is still posting tips for his web page, but this would be a cool one to add if there is a neat solution. Maybe he already has it there.)
>
> Gordon

Subject: Re: Function call string
Posted by David Fanning on Fri, 10 Apr 2015 13:09:35 GMT
View Forum Message <> Reply to Message

Gordon Farquharson writes:

>
> Is there a way to access the string that was used to call a function or procedure in IDL? For example, lets say I call:
>
> display, 'A carrot is orange'
>

> Is there a way to access the string "display, 'A carrot is orange'" from within the display procedure?
>
> The reason I'd like to do this is that I'd like to record how a function or procedure was called in the meta data that the function or procedure generates, so that it is easy to make the same call again in the future.
>
> (I seem to recall that there was a discussion on this topic previously, but I can't find the thread.)
>
> I realize that I could construct the function call string from the arguments, but the output is not generic or pretty because one has to decide on things like how many decimal places to use to represent floating point values, and one has to take care of each keyword setting.
>
> Maybe a better self-documenting way to do this is to have a configuration file, e.g., an XML file that the procedure or function reads, and that accompanies the output. While I have done this previously for other routines, it seems a little overkill for my current application. Maybe it is the best solution though.
>
> (Not sure if David is still posting tips for his web page, but this would be a cool one to add if there is a neat solution. Maybe he already has it there.)

In a rudimentary way (not the elegant way you describe), this idea of "saving a command" is at the heart of the entire Coyote Graphics Library, since this is exactly what needs to be done to create a resizeable graphics window with old graphics commands. See the cgCmdWindow object code for how it is done.

I couldn't figure out how to make the Coyote system completely generic to any and all commands with the "old" IDL commands I wanted to remain faithful to, so this works only within the limited universe of graphics commands. But, I think with the creative use of some of IDL's new language features, that something like this could now be written generically.

For example, one limitation in the Coyote Graphics System is the inability to create (or use) an output keyword. I read not too long ago on the IDL Blog in the ExelisVis site about a convoluted way to make this possible with hashes. It's not simple, but it is probably possible.

I force the user to specify the "command" as a string. But, it is easy enough to get the name of the program module that called another. You can use cgWhoCalledMe from the Coyote Library, for example.

http://www.idlcoyote.com/programs/cgwhocalledme.pro

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: Function call string
Posted by Gordon Farquharson on Fri, 10 Apr 2015 15:51:20 GMT
View Forum Message <> Reply to Message

On Thursday, April 9, 2015 at 10:30:22 PM UTC-7, Paulo Penteado wrote:

> pro functioncallstring,arg1,arg2,key1=key1,key2=key2
> help,/recall,output=o
> print,o[1]
> end

Thanks for the tip. This method works for what I want in this case. I wasn't aware that help provides the command line history.

Gordon

Subject: Re: Function call string
Posted by Gordon Farquharson on Fri, 10 Apr 2015 16:22:49 GMT
View Forum Message <> Reply to Message

Hi David

On Friday, April 10, 2015 at 6:09:41 AM UTC-7, David Fanning wrote:

> In a rudimentary way (not the elegant way you describe), this idea of
> "saving a command" is at the heart of the entire Coyote Graphics
> Library, since this is exactly what needs to be done to create a
> resizeable graphics window with old graphics commands. See the
> cgCmdWindow object code for how it is done.

Thanks for the pointer to cgCmdWindow. Now that you say it, of course the CG routines reconstruct the function call - some part of my brain had always realized that, but I completely missed the connection to my current problem.

I'll take a look at cgCmdWindow when I get a chance. For now, Paulo's solutions works for what I need. I did find the following thread that might also be useful in the future.

https://groups.google.com/d/msg/comp.lang.idl-pvwave/vj7IIy5 cVlY/eJqfTh5w_GsJ