Subject: Re: .trace not working?
Posted by BLemire@ittvis.com on Wed, 05 Aug 2009 21:16:58 GMT
View Forum Message <> Reply to Message

On Aug 5, 2:13 pm, Bruce Bowler <bbow...@bigelow.org> wrote:
> Environment IDL 7.0 (and 7.1) on fedora 10 (and fedora 11), 32 bit version
>
> Imagine, if you will, a simple .pro file such as the following (never
> mind that this doesn't really do anything)...
>
> pro test1
> x = randomn(seed1, 16)
> y = randomn(seed2, 16)
> plot, x, y
> end
>
> Now imagine that you fire up IDL and type the following commands at the
> IDL> prompt
>
> breakpoint,"test1.pro",2
> test1
>
> and further, attempt to stretch you imagination to the point that, when
> you get to the IDL prompt as a result of the breakpoint, you type
>
> .trace
>
> You, of course, type that particular command because the documentation
> says, and I quote,
>
> .TRACE - Similar to .CONTINUE, but displays each line of code
> before execution.
>
> but when you hit the return key at the end of .trace, you see no
> additional outputting of code prior to executing each line of code.
>
> Where have you gone wrong???
>
> Bruce
>
> --
> +------------------+------------------------------------ ------------+
> Bruce Bowler | I am free of all prejudices; I hate everyone
> 1.207.633.9600 | equally. - W. C. Fields
> bbow...@bigelow.org |
> +------------------+------------------------------------ ------------+

Bruce,

Based on the documentation in IDL 70 and IDL 71 the following description is provided for the .TRACE command:

"The .TRACE command continues execution of a program that has stopped because of an error, a stop statement, or a keyboard interrupt."

This definition was located in

IDL API Reference Guides > IDL Reference Guide > Part I: IDL Command Reference > Dot Commands

This documentation is different than that which would be available in printed form. The change from the definition you had to the new definition occurred in IDL 4.0.1 for exactly the reason you provided. I hope that this helps.

Based on the implementation of the .TRACE command, it is only supposed to complete executing the code. I was able to get the code to complete as expected. It might be helpful if a verbose keyword was added to the .TRACE command so that each line of code was output prior to execution.

Best Regards,

Brandon

Subject: Re: .trace not working?
Posted by Michael Galloy on Thu, 06 Aug 2009 00:39:16 GMT
View Forum Message <> Reply to Message

BLemire@ittvis.com wrote:
> On Aug 5, 2:13 pm, Bruce Bowler <bbow...@bigelow.org> wrote:
>> Environment IDL 7.0 (and 7.1) on fedora 10 (and fedora 11), 32 bit version
>>
>> Imagine, if you will, a simple .pro file such as the following (never
>> mind that this doesn't really do anything)...
>>
>> pro test1
>> x = randomn(seed1, 16)
>> y = randomn(seed2, 16)
>> plot, x, y
>> end
>>
>> Now imagine that you fire up IDL and type the following commands at the
>> IDL> prompt
>>

>> breakpoint,"test1.pro",2
>> test1
>>
>> and further, attempt to stretch you imagination to the point that, when
>> you get to the IDL prompt as a result of the breakpoint, you type
>>
>> .trace
>>
>> You, of course, type that particular command because the documentation
>> says, and I quote,
>>
>> .TRACE - Similar to .CONTINUE, but displays each line of code
>> before execution.
>>
>> but when you hit the return key at the end of .trace, you see no
>> additional outputting of code prior to executing each line of code.
>>
>> Where have you gone wrong???
>>
>> Bruce
>>
>> --
>> +------------------+-------------------------------------- ------------+
>> Bruce Bowler | I am free of all prejudices; I hate everyone
>> 1.207.633.9600 | equally. - W. C. Fields
>> bbow...@bigelow.org |
>> +------------------+-------------------------------------- ------------+
>
> Bruce,
>
> Based on the documentation in IDL 70 and IDL 71 the following
> description is provided for the .TRACE command:
>
> "The .TRACE command continues execution of a program that has stopped
> because of an error, a stop statement, or a keyboard interrupt."
>
> This definition was located in
>
> IDL API Reference Guides > IDL Reference Guide > Part I: IDL Command
> Reference > Dot Commands
>
> This documentation is different than that which would be available in
> printed form. The change from the definition you had to the new
> definition occurred in IDL 4.0.1 for exactly the reason you provided.
> I hope that this helps.
>
> Based on the implementation of the .TRACE command, it is only supposed
> to complete executing the code. I was able to get the code to

> complete as expected. It might be helpful if a verbose keyword was
> added to the .TRACE command so that each line of code was output prior
> to execution.

So .trace is exactly equivalent to .continue?

Mike
--
www.michaelgalloy.com
Associate Research Scientist
Tech-X Corporation

Subject: Re: .trace not working?
Posted by Bruce Bowler on Thu, 06 Aug 2009 12:13:57 GMT
View Forum Message <> Reply to Message

On Wed, 05 Aug 2009 14:16:58 -0700, BLemire@ittvis.com wrote:

> On Aug 5, 2:13 pm, Bruce Bowler <bbow...@bigelow.org> wrote:
>> Environment IDL 7.0 (and 7.1) on fedora 10 (and fedora 11), 32 bit
>> version
>>
>> Imagine, if you will, a simple .pro file such as the following (never
>> mind that this doesn't really do anything)...
>>
>> pro test1
>> x = randomn(seed1, 16)
>> y = randomn(seed2, 16)
>> plot, x, y
>> end
>>
>> Now imagine that you fire up IDL and type the following commands at the
>> IDL> prompt
>>
>> breakpoint,"test1.pro",2
>> test1
>>
>> and further, attempt to stretch you imagination to the point that, when
>> you get to the IDL prompt as a result of the breakpoint, you type
>>
>> .trace
>>
>> You, of course, type that particular command because the documentation
>> says, and I quote,
>>
>> .TRACE - Similar to .CONTINUE, but displays each line of
>> code

>> before execution.
>>
>> but when you hit the return key at the end of .trace, you see no
>> additional outputting of code prior to executing each line of code.
>>
>> Where have you gone wrong???
>>
>
> Bruce,
>
> Based on the documentation in IDL 70 and IDL 71 the following
> description is provided for the .TRACE command:
>
> "The .TRACE command continues execution of a program that has stopped
> because of an error, a stop statement, or a keyboard interrupt."
>
> This definition was located in
>
> IDL API Reference Guides > IDL Reference Guide > Part I: IDL Command
> Reference > Dot Commands
>
> This documentation is different than that which would be available in
> printed form. The change from the definition you had to the new
> definition occurred in IDL 4.0.1 for exactly the reason you provided. I
> hope that this helps.
>
> Based on the implementation of the .TRACE command, it is only supposed
> to complete executing the code. I was able to get the code to complete
> as expected. It might be helpful if a verbose keyword was added to the
> .TRACE command so that each line of code was output prior to execution.
>
> Best Regards,
>
> Brandon

2 things...

First, it's worth noting, the text I quoted is from idlhelp that shipped
with idl 7.0.

From the top of the idlhelp window,

IDL API Reference Guides > IDL Reference Guide > Part I: IDL Command
Reference > Functional List of IDL Routines (and then select debugging)

Second, You state "for exactly the reason you provided", but I didn't
provide a reason, I provided a question. Why doesn't trace work as

documented (or to put in Mike's terms, why have .trace if it is now
exactly like .continue?)

Is there a *SUPPORTED* (hell, at this point I'd take unsupported) way to
TRACE the execution of a routine?

Bruce

--
+------------------+------------------------------------ ------------+
Bruce Bowler | Honor thy father and thy mother, for they shall be
1.207.633.9600 | interviewed. - Emo Phillips
bbowler@bigelow.org |
+------------------+------------------------------------ ------------+

Subject: Re: .trace not working?
Posted by David Fanning on Thu, 06 Aug 2009 12:18:48 GMT
View Forum Message <> Reply to Message

Bruce Bowler writes:

> Is there a *SUPPORTED* (hell, at this point I'd take unsupported) way to
> TRACE the execution of a routine?

Lots of PRINT statements! ;-)

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: .trace not working?
Posted by Bruce Bowler on Thu, 06 Aug 2009 13:44:35 GMT
View Forum Message <> Reply to Message

On Thu, 06 Aug 2009 06:18:48 -0600, David Fanning wrote:

> Bruce Bowler writes:
>
>> Is there a *SUPPORTED* (hell, at this point I'd take unsupported) way
>> to TRACE the execution of a routine?
>
> Lots of PRINT statements! ;-)
>
> Cheers,
>
> David

I figured you'd say that :-). There are about 1000 lines of code (lots
of different routines) where my procedure could be stopping. I'd really
rather not add another 1000 print statements :-)

Oh well... sore fingers, here I come.

Bruce

--
+------------------+------------------------------------ ------------+
Bruce Bowler | Patriotism is the last refuge of a scoundrel. -
1.207.633.9600 | Samuel Adams
bbowler@bigelow.org |
+------------------+------------------------------------ ------------+

Subject: Re: .trace not working?
Posted by jkj on Fri, 07 Aug 2009 00:52:39 GMT
View Forum Message <> Reply to Message

On Aug 6, 8:44 am, Bruce Bowler <bbow...@bigelow.org> wrote:
> On Thu, 06 Aug 2009 06:18:48 -0600, David Fanning wrote:
>> Bruce Bowler writes:
>
>>> Is there a *SUPPORTED* (hell, at this point I'd take unsupported) way
>>> to TRACE the execution of a routine?
>
>> Lots of PRINT statements! ;-)
>
>> Cheers,
>
>> David
>
> I figured you'd say that :-). There are about 1000 lines of code (lots
> of different routines) where my procedure could be stopping. I'd really
> rather not add another 1000 print statements :-)

>
> Oh well... sore fingers, here I come.
>
> Bruce
>
> --
> +------------------+------------------------------------ ------------+
> Bruce Bowler | Patriotism is the last refuge of a scoundrel. -
> 1.207.633.9600 | Samuel Adams
> bbow...@bigelow.org |
> +------------------+------------------------------------ ------------+

I have always used ".cont" to resume execution and when I want to know the nesting of calls by which I arrived at the given stop I enter "blah", giving me a traceable path up through the calling code.

It certainly sounds like ".trace" is exactly equivalent to ".continue" but nobody wants to address it clearly - weird. I can't imagine why IDL supplies ".trace" and ".continue" without giving a clear reason about the differences - .trace is clearly misnamed if all it does is continue execution... must be something more with a bit deeper digging, but I'm not getting my shovel out for that one.

For object code I create a function call for each object "::get_hack_val" that returns a '1' when I want to trace execution paths, so every routine checks "->get_hack_val()" and if it returns "1" then a print of the routine's name and any other helpful information (status of keywords, for example) is printed.

Subject: Re: .trace not working?
Posted by David Fanning on Fri, 07 Aug 2009 01:40:42 GMT
View Forum Message <> Reply to Message

jkj writes:

> I have always used ".cont" to resume execution and when I want to know
> the nesting of calls by which I arrived at the given stop I enter
> "blah", giving me a traceable path up through the calling code.
>
> It certainly sounds like ".trace" is exactly equivalent to ".continue"
> but nobody wants to address it clearly - weird. I can't imagine why
> IDL supplies ".trace" and ".continue" without giving a clear reason
> about the differences - .trace is clearly misnamed if all it does is
> continue execution... must be something more with a bit deeper
> digging, but I'm not getting my shovel out for that one.
>
> For object code I create a function call for each object

> "::get_hack_val" that returns a '1' when I want to trace execution
> paths, so every routine checks "->get_hack_val()" and if it returns
> "1" then a print of the routine's name and any other helpful
> information (status of keywords, for example) is printed.

It is always interesting to me to see how people debug
their code. I'm always most interested in the people who
don't read their error messages. They seem almost a different
breed to me.

But if you get to a stop, and you want to know how you
got there, I don't see how "blah" can help. I would
think HELP, /TRACEBACK might do more good. :-)

I debug in the Workbench, of course, so when I come to
a stop I usually just step over routines to the next
line of code (F6) or step into routines (F5). Or, if
I want to run awhile, just put my cursor on the line
I want to run to and type CNTL-R. I don't usually feel
a need for anything fancier than that.

Debugging mostly involves what goes on between your
ears, and it especially helps if you are not closely
wedded to your own ideas of how you expect things to
be. A certain ability to not take yourself too
seriously is what makes a good debugger, I think.

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: .trace not working?
Posted by penteado on Fri, 07 Aug 2009 17:44:51 GMT
View Forum Message <> Reply to Message

On Aug 6, 10:40 pm, David Fanning <n...@dfanning.com> wrote:
> I debug in the Workbench, of course, so when I come to
> a stop I usually just step over routines to the next
> line of code (F6) or step into routines (F5). Or, if
> I want to run awhile, just put my cursor on the line

> I want to run to and type CNTL-R. I don't usually feel
> a need for anything fancier than that.

Same here. The workbench also has the debug view where I can see the call stack, to know where the routines were called from (similar to help,/traceback, but it is always shown and updated), and to move to different scopes without moving the point where execution is stopped. I can browse variable values in the variable view, or by just parking the cursor over a variable name in the editor. And in the workbench, console references to lines in the source code are links to show them in the editor.

And yet I still frequently see people using just the command line, editing source files in vi or emacs.

Subject: Re: .trace not working?
Posted by JDS on Fri, 07 Aug 2009 20:11:37 GMT
View Forum Message <> Reply to Message

> And yet I still frequently see people using just the command line,
> editing source files in vi or emacs.

Ho-now. In Emacs with IDLWAVE I can:

1) report on the call stack at a stopped location.
2) use single keystrokes to examine variables (not just print, but view structure fields, see widget dimensions etc., or other things you can dream up and easily configure). I can also drag out or use shortcuts to examine any arbitrary expressions in the code (like '(x + y^z)').
3) move up and down through the call stack, examining variables or expressions in any parent scope (in practice I just do this rather than read the traceback).
4) trivially set, remove, or alter breakpoints, including conditional and repeat breakpoints
5) continue, step, stop, continue to the line at the cursor point, continue up and out of the enclosing block (for loop, etc.)

So basically (as far as I can tell) every debugging option in the Workbench, and a bit more. Not as much clicking, but for some that's an advantage. Did I mention it's been doing this for about 10 years?

Subject: Re: .trace not working?
Posted by David Fanning on Fri, 07 Aug 2009 20:36:13 GMT

JDS writes:

>
>> And yet I still frequently see people using just the command line,
>> editing source files in vi or emacs.
>
> Ho-now. In Emacs with IDLWAVE I can:
>
> 1) report on the call stack at a stopped location.
> 2) use single keystrokes to examine variables (not just print, but
> view structure fields, see widget dimensions etc., or other things you
> can dream up and easily configure). I can also drag out or use
> shortcuts to examine any arbitrary expressions in the code (like '(x +
> y^z)').
> 3) move up and down through the call stack, examining variables or
> expressions in any parent scope (in practice I just do this rather
> than read the traceback).
> 4) trivially set, remove, or alter breakpoints, including conditional
> and repeat breakpoints
> 5) continue, step, stop, continue to the line at the cursor point,
> continue up and out of the enclosing block (for loop, etc.)
>
> So basically (as far as I can tell) every debugging option in the
> Workbench, and a bit more. Not as much clicking, but for some that's
> an advantage. Did I mention it's been doing this for about 10
> years?

If I had had better parents, they would have provided
an EMACS editor for me in the crib, and I wouldn't be such
a miserable person today. But EMACS is sort of like tennis.
Not really the kind of thing you are going to be any good
at if you take it up after, say, the age of 30. Just too
many good years of muscle memory destroyed by the beer,
I guess. :-(

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: .trace not working?
Posted by penteado on Sat, 08 Aug 2009 03:06:35 GMT
View Forum Message <> Reply to Message

On Aug 7, 5:11 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:
> Ho-now. In Emacs with IDLWAVE I can:
>
> 1) report on the call stack at a stopped location.
> 2) use single keystrokes to examine variables (not just print, but
> view structure fields, see widget dimensions etc., or other things you
> can dream up and easily configure). I can also drag out or use
> shortcuts to examine any arbitrary expressions in the code (like '(x +
> y^z)').
> 3) move up and down through the call stack, examining variables or
> expressions in any parent scope (in practice I just do this rather
> than read the traceback).
> 4) trivially set, remove, or alter breakpoints, including conditional
> and repeat breakpoints
> 5) continue, step, stop, continue to the line at the cursor point,
> continue up and out of the enclosing block (for loop, etc.)
>
> So basically (as far as I can tell) every debugging option in the
> Workbench, and a bit more. Not as much clicking, but for some that's
> an advantage. Did I mention it's been doing this for about 10
> years?

Ok. I did not know one could do that much in IDL with Emacs, and I am also impressed that it has been the case for so long. I have for some time intended to examine Emacs in more detail, this encourages me to do it sooner.

However, the people I was referring to, which I frequently see, do not use such features, they only edit source files as one would do in vi or any other simple text editor, run things in IDL in a command line, and fill the code with prints when trying to debug, with a lot of avoidable suffering.

Not to try to make it a contest, just to inform more to others (as you have informed me of what can be done in Emacs): in the workbench editor it is also possible to see a routine's arguments when you park the cursor over its name (even when it is user-defined), open its help, and jump to its definition (even when in another file). There is help on method names and structure fields when typing them in the editor and command line, and there are browsers for the command history, profiler results, and routines defined in a file (the outline view).

Subject: Re: .trace not working?
Posted by Jeremy Bailin on Sun, 09 Aug 2009 02:19:37 GMT
View Forum Message <> Reply to Message

On Aug 7, 11:06 pm, pp <pp.pente...@gmail.com> wrote:
> On Aug 7, 5:11 pm, JDS <jdtsmith.nos...@yahoo.com> wrote:
>
>
>
>
>
>
>> Ho-now. In Emacs with IDLWAVE I can:
>
>> 1) report on the call stack at a stopped location.
>> 2) use single keystrokes to examine variables (not just print, but
>> view structure fields, see widget dimensions etc., or other things you
>> can dream up and easily configure). I can also drag out or use
>> shortcuts to examine any arbitrary expressions in the code (like '(x +
>> y^z)').
>> 3) move up and down through the call stack, examining variables or
>> expressions in any parent scope (in practice I just do this rather
>> than read the traceback).
>> 4) trivially set, remove, or alter breakpoints, including conditional
>> and repeat breakpoints
>> 5) continue, step, stop, continue to the line at the cursor point,
>> continue up and out of the enclosing block (for loop, etc.)
>
>> So basically (as far as I can tell) every debugging option in the
>> Workbench, and a bit more. Not as much clicking, but for some that's
>> an advantage. Did I mention it's been doing this for about 10
>> years?
>
> Ok. I did not know one could do that much in IDL with Emacs, and I am
> also impressed that it has been the case for so long. I have for some
> time intended to examine Emacs in more detail, this encourages me to
> do it sooner.
>
> However, the people I was referring to, which I frequently see, do not
> use such features, they only edit source files as one would do in vi
> or any other simple text editor, run things in IDL in a command line,
> and fill the code with prints when trying to debug, with a lot of
> avoidable suffering.
>
> Not to try to make it a contest, just to inform more to others (as you
> have informed me of what can be done in Emacs): in the workbench
> editor it is also possible to see a routine's arguments when you park
> the cursor over its name (even when it is user-defined), open its
> help, and jump to its definition (even when in another file). There is
> help on method names and structure fields when typing them in the

> editor and command line, and there are browsers for the command
> history, profiler results, and routines defined in a file (the outline
> view).

Ah, but at the expense of running the workbench. ;-) To each their own poison!

-Jeremy.

Subject: Re: .trace not working?
Posted by BLemire@ittvis.com on Sun, 09 Aug 2009 03:14:28 GMT
View Forum Message <> Reply to Message

Bruce and all others,

I apologize for not finding the definition that you provided and now see where I was mistaken. I will follow up by asking what .TRACE was intended to do and making sure to put in a request to have it perform as expected. It appears that the information provided on the debugging page differs from that on the .TRACE page. I will let you all know what I find out.

Best Regards,

Brandon

Subject: Re: .trace not working?
Posted by JDS on Mon, 10 Aug 2009 17:33:51 GMT
View Forum Message <> Reply to Message

> Not to try to make it a contest, just to inform more to others (as you
> have informed me of what can be done in Emacs): in the workbench
> editor it is also possible to see a routine's arguments when you park
> the cursor over its name (even when it is user-defined), open its
> help, and jump to its definition (even when in another file). There is
> help on method names and structure fields when typing them in the
> editor and command line, and there are browsers for the command
> history, profiler results, and routines defined in a file (the outline
> view).

I haven't really used the workbench much, so this is interesting to know. Routine info is in IDLWAVE for system-distributed, compiled, buffer-loaded, and scanned routines. You can also pre-scan all your local routines, and many libraries (like Fanning, NASAlib, etc.) ship pre-scanned. Which means you can get help on a routine even before

IDL has compiled it, even if you can't quite remember its name, even if the shell isn't running (e.g. no license available). Context-sensitive help is there too. You can also jump directly to help on individual keywords, saving you scrolling down to find them. You can get help on structure tags for structures defined elsewhere in your code (like 'state'), object fields, control statements, system variables, etc., etc.

That said, completion is probably the one area where IDLWAVE still (AFAIK) exceeds the Workbench: your can complete routine names, keywords, reserved words, system variables, structure members, object methods, filenames on disk, etc., all with the (alt-)tab key. I find myself hitting tab and groaning constantly when I've used the Workbench. But, as David says, admittedly the learning curve is steeper compared to the point-and-click browsers. Actually, it's not that it's really that steep, it's just that it sits near the top of the giant Emacs learning peak.

All of this brings up an

++++++++++++++++++++++++++++
Informal survey:

( ) IDLWHO?
( ) I recently switched from IDLWAVE to the Workbench.
( ) I use both IDLWAVE and Workbench, depending on the setting/task/context.
( ) You can pry IDLWAVE from my cold, dead hands.
++++++++++++++++++++++++++++

For the record, I don't mind in the least if people do switch to the WB. It's the first really featureful IDE they've offered.

JD
