Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Tue, 12 Oct 2010 18:00:07 GMT
View Forum Message <> Reply to Message

Wayne Landsman writes:

> I'm not sure if this IDL 8.0 bug has been mentioned. (At least I
> think it is a bug -- perhaps someone can convince me that it is a
> feature.)

Oh, my goodness! What in the world are they doing to IDL!?

Cheers,

David

P.S. I'm all for progress and new things, but when every
new version is two steps backwards, you really begin to
wonder what you are paying for. The documentation is screwed up
anyway, so even if you knew where the error occurred, you
probably still couldn't debug your programs. Maybe with
the new graphics, you don't have to. :-)

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

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Tue, 12 Oct 2010 18:11:07 GMT
View Forum Message <> Reply to Message

On Oct 12, 2:35 pm, wlandsman <wlands...@gmail.com> wrote:
> I'm not sure if this IDL 8.0 bug has been mentioned. (At least I
> think it is a bug -- perhaps someone can convince me that it is a
> feature.)

I had not noticed this before. It seems to only happen when 'on_error,
2' is used.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by chris_torrence@NOSPAM on Tue, 12 Oct 2010 19:45:39 GMT
View Forum Message <> Reply to Message

On Oct 12, 12:11 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Oct 12, 2:35 pm, wlandsman <wlands...@gmail.com> wrote:
>
>> I'm not sure if this IDL 8.0 bug has been mentioned. (At least I
>> think it is a bug -- perhaps someone can convince me that it is a
>> feature.)
>
> I had not noticed this before. It seems to only happen when 'on_error,
> 2' is used.

Hi all,
This is by design. When developing the new graphics, we noticed that many of the error messages had overly-long stack traces, because on_error,2 always dumped out the stack trace from where the error message was triggered. We changed it in IDL 8.0, so that now it only prints out the stack trace from where IDL actually stops execution. For on_error,2 this is the "caller of the program unit that called ON_ERROR".

The general philosophy is that on_error,2 should be used for "library" routines, where the caller should not need to care about the internal workings of the library. If you are writing your own routines, or are debugging an existing routine, then I would recommend that you disable the on_error,2 command until you have completed your routine and are ready to "release" it. I think the IDL help for ON_ERROR mentions this.

Two points:
* There was a bug in IDL 8.0, where this change also affected on_error,1. This has been fixed in the upcoming patch - on_error,1 will again behave the way it did before.
* This change in behavior was unfortunately left out of the release notes and the documentation. This has been corrected.

Hope this hasn't caused too much confusion.

Cheers,
Chris
ITTVIS

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Tue, 12 Oct 2010 19:58:23 GMT
View Forum Message <> Reply to Message

Chris Torrence writes:

> This is by design. When developing the new graphics, we noticed that

> many of the error messages had overly-long stack traces, because
> on_error,2 always dumped out the stack trace from where the error
> message was triggered. We changed it in IDL 8.0, so that now it only
> prints out the stack trace from where IDL actually stops execution.
> For on_error,2 this is the "caller of the program unit that called
> ON_ERROR".
>
> The general philosophy is that on_error,2 should be used for "library"
> routines, where the caller should not need to care about the internal
> workings of the library. If you are writing your own routines, or are
> debugging an existing routine, then I would recommend that you disable
> the on_error,2 command until you have completed your routine and are
> ready to "release" it. I think the IDL help for ON_ERROR mentions
> this.

No, no, no. This is NOT what I want at all! Oh, dear.
Twenty years of error handling down the drain.

I'm going to have to think about this some, but my
first reaction is that this is NOT a good idea. :-(

The source of errors when working with objects (as I am
sure you know as well as I do) are often a LONG way away
from where the error manifests itself. I think this has
the possibility of making debugging impossibly hard.

Oh, dear!

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

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Tue, 12 Oct 2010 20:54:10 GMT
View Forum Message <> Reply to Message

On Oct 12, 4:45 pm, Chris Torrence <gorth...@gmail.com> wrote:
> This is by design. When developing the new graphics, we noticed that
> many of the error messages had overly-long stack traces, because
> on_error,2 always dumped out the stack trace from where the error
> message was triggered. We changed it in IDL 8.0, so that now it only
> prints out the stack trace from where IDL actually stops execution.
> For on_error,2 this is the "caller of the program unit that called
> ON_ERROR".

I was wondering if this was the case, since I agree this was desirable. But because I saw no mention to it in the help, and I happened to misread the test case above, I thought something could be wrong.

Not showing the full trace is consistent with returning to the caller. That is, the code below behaves as I expected, and agree that should be the case:

```
function test1
On_error,2
a = bindgen(32)
c = long(a,30,1)
return,1
end

pro test
print,test1()
return
end

IDL> test
% Compiled module: TEST.
% Specified offset to array is out of range: A.
% Execution halted at: TEST             9 /home/penteado/idl/
test.pro
%                $MAIN$
```

As the trace shows where the place where the routine with 'on_error,2' was called.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by Bill Nel on Tue, 12 Oct 2010 21:49:05 GMT

View Forum Message <> Reply to Message

On Oct 12, 3:58 pm, David Fanning <n...@dfanning.com> wrote:
> Chris Torrence writes:
>> This is by design.

1) The error messages IDL returns are pretty useless unless one knows which routine generated the error. You might as well have IDL return "Something's wrong."

2) I use On_Error, 2 and Message, ... in "mature" code to report errors from parameter checking -- very handy for interactive use. The

new behavior forces me to give this up or to waste time (considerably)
tracking down a bug when it does appear.

3) Why don't you just attach the routine name to IDL's error message?

--Wayne

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by Paul Van Delst[1] on Tue, 12 Oct 2010 22:16:58 GMT
View Forum Message <> Reply to Message

Paulo Penteado wrote:
> On Oct 12, 4:45 pm, Chris Torrence <gorth...@gmail.com> wrote:
>> This is by design. When developing the new graphics, we noticed that
>> many of the error messages had overly-long stack traces, because
>> on_error,2 always dumped out the stack trace from where the error
>> message was triggered. We changed it in IDL 8.0, so that now it only
>> prints out the stack trace from where IDL actually stops execution.
>> For on_error,2 this is the "caller of the program unit that called
>> ON_ERROR".
>
> I was wondering if this was the case, since I agree this was
> desirable. But because I saw no mention to it in the help, and I
> happened to misread the test case above, I thought something could be
> wrong.
>
> Not showing the full trace is consistent with returning to the caller.

<disclaimer>
I do not use ON_ERROR for error handling. I always use CATCH.
</disclaimer>

Not sure if I agree with the consistency you mention. Since the error handler prints out a message it should at least
provide the name of the procedure/function in which the error occurred. Without it, there is zero context for the error.

But, one thing I don't think is good programming practice in general is the advice provided in the IDL help for ON_ERROR:

<quote>
This form of error recovery makes debugging a routine difficult because the routine is exited as soon as an error
occurs; therefore, [ON_ERROR,2] should be added once the code is completely tested.
</quote>

which, to me, translates to:

Completely test your code. Then change it.

I'm a notorious fat-finger-er when it comes to typing so once I've completely tested something, I'm rather loathe to
change it. Particularly when the part being changed is the error handling....which may or may not be able to be fully
tested? I'm somewhat confused (and probably paranoid about the subject :o).

Maybe people should start switching to using CATCH instead for their day-to-day error handling?

cheers,

paulv

> That is, the code below behaves as I expected, and agree that should
> be the case:
>
> function test1
> On_error,2
> a = bindgen(32)
> c = long(a,30,1)
> return,1
> end
>
> pro test
> print,test1()
> return
> end
>
> IDL> test
> % Compiled module: TEST.
> % Specified offset to array is out of range: A.
> % Execution halted at: TEST 9 /home/penteado/idl/
> test.pro
> % $MAIN$
>
> As the trace shows where the place where the routine with 'on_error,2'
> was called.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Tue, 12 Oct 2010 22:25:32 GMT

View Forum Message <> Reply to Message

On Oct 12, 7:16 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
>> Not showing the full trace is consistent with returning to the caller.
>

> <disclaimer>
> I do not use ON_ERROR for error handling. I always use CATCH.
> </disclaimer>
>
> Not sure if I agree with the consistency you mention. Since the error handler prints out a message it should at least
> provide the name of the procedure/function in which the error occurred. Without it, there is zero context >for the error.

My view is that setting 2 for on_error is already asking to be thrown
out of the context of the error. That is why I see it as consistent.
It is not zero information on where the error happens, because you get
the trace up to the point where the routine with 'on_error,2' is
called.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Tue, 12 Oct 2010 22:51:42 GMT
View Forum Message <> Reply to Message

Chris Torrence writes:

> This is by design. When developing the new graphics, we noticed that
> many of the error messages had overly-long stack traces, because
> on_error,2 always dumped out the stack trace from where the error
> message was triggered. We changed it in IDL 8.0, so that now it only
> prints out the stack trace from where IDL actually stops execution.
> For on_error,2 this is the "caller of the program unit that called
> ON_ERROR".

This turns out not be as big a problem for me as I thought
it might be. The method I use to catch most errors in my
programs:

```
Catch, theError
IF theError NE 0 THEN BEGIN
   Catch, /Cancel
   void = Error_Message()
   RETURN
END
```

still seems to work normally with the "new" ON_ERROR
behavior. Error_Message prints out the proper trace to the
error.

> The general philosophy is that on_error,2 should be used for "library"
> routines, where the caller should not need to care about the internal
> workings of the library. If you are writing your own routines, or are

> debugging an existing routine, then I would recommend that you disable
> the on_error,2 command until you have completed your routine and are
> ready to "release" it. I think the IDL help for ON_ERROR mentions
> this.

I'm not sure who's "general philosophy" we are talking about here, but I would say my "general philosophy" is not to change the way software works unless there is some extremely compelling reason for it. Maybe scaring the bejesus out of customers with long error messages from over-complicated software is a compelling reason. I couldn't say.

But, general philosophy be damned, a lot of people rely on error handling to remain the same from one version of IDL to the next. I would say that is a reasonable expectation.

It's one thing to be the 500-pound gorilla and stomp all over established IDL code when you decide to name your programs. It's something else to make changes that break a lot of established code. Can anyone *really* think this is good for business? Even *new* business?

I still think this change is a mistake.

Maybe this should be implemented as a new keyword:

```
On_Error, 2, /POLYANNA_MESSAGE
On_Error, 2, /BRUTAL_TRUTH
```

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: IDL 8.0 bug -- line number of errors not given
Posted by wlandsman on Wed, 13 Oct 2010 02:42:14 GMT
View Forum Message <> Reply to Message

On Oct 12, 6:16 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:

> Maybe people should start switching to using CATCH instead for their day-to-day error handling?

OK, here's my simpleton reason for not using CATCH. If I use ON_ERROR I just add 1 line of code at the beginning of each procedure

```
On_error, 2
```

But if I use CATCH I need to add a whole paragraph somewhere (where?) in the code

```
Catch, theError
IF theError NE 0 THEN BEGIN
   Catch, /Cancel
   void = Error_Message()
   RETURN
END
```

So what are the advantages of using CATCH? (This is not a trick question -- I've had a mental block about how to use CATCH.) --Wayne

---

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 02:46:13 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> I was wondering if this was the case, since I agree this was
> desirable.

Humm. In what sense of the word do you find this "desirable"?
>
> Not showing the full trace is consistent with returning to the caller.

Returning to the caller for what purpose?

> That is, the code below behaves as I expected, and agree that should
> be the case:

Why is this what you expected? Because of your experience with other software? What other software acts like this? You thought IDL's behavior in the past was inconsistent with good programming practices?
>

```
> function test1
> On_error,2
> a = bindgen(32)
> c = long(a,30,1)
> return,1
```

> end
>
> pro test
> print,test1()
> return
> end
>
> IDL> test
> % Compiled module: TEST.
> % Specified offset to array is out of range: A.
> % Execution halted at: TEST 9 /home/penteado/idl/
> test.pro
> % $MAIN$
>
> As the trace shows where the place where the routine with 'on_error,2'
> was called.

Well, I agree this is the place it was called. But what possible
good is this information? How would you use this information?

Why would you use ON_ERROR, 2? Should it be eliminated from
the language? How would you use it now that it is working
as you expect it to?

I truly do not understand how this can be a good thing.
I'm all ears.

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: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 03:02:44 GMT
View Forum Message <> Reply to Message

Wayne Landsman writes:

> OK, here's my simpleton reason for not using CATCH. If I use
> ON_ERROR I just add 1 line of code at the beginning of each procedure
>

>   On_error, 2
>
> But if I use CATCH I need to add a whole paragraph somewhere (where?)
> in the code
>
>   Catch, theError
>   IF theError NE 0 THEN BEGIN
>   Catch, /Cancel
>   void = Error_Message()
>   RETURN
>   END

You add it to the code just before the part of the code that
is going to generate the error. (The very first line in most
people's code!) I usually add it as the first piece of code after
Compile_Opt idl2.

> So what are the advantages of using CATCH? (This is not a trick
> question -- I've had a mental block about how to use CATCH.) --Wayne

One of the advantages is that it catches all kinds of errors
(I/O, run-time, etc.) Another is that you can put catches anywhere
you anticipate an error occurring, and you can even fix errors and
continue program execution. A HUGE advantage in widget programs is
that you can keep widget programs running and alive, even when errors
occur. (This is occasionally a BAD thing, but more often than not a
GOOD thing.)

I use Catches because I can combine them with Error_Message to
get nicely formatted error results and user notification of errors.
The traceback information is accurate and complete (even in IDL 8
in the cases I've tested so far, and even when used to catch errors
coming from ON_ERROR,2 conditions in other modules).

The most compelling reason, of course, is that it will soon be
the only way to find errors in many IDL programs! ;-)

Cheers,

David

P.S. Let's just say I am beginning to miss those idyllic days
of silent error handlers in iTools!

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by wlandsman on Wed, 13 Oct 2010 03:34:47 GMT
View Forum Message <> Reply to Message

On Oct 12, 4:54 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:

> Not showing the full trace is consistent with returning to the caller.

Perhaps there are two issues here. I can agree with not showing the full trace when an error occurs inside a procedure with ON_ERROR,2 set. But an error message is still being displayed

% Specified offset to array is out of range: A.

As Bil Nei and others have mentioned - why not include the line number and at least the procedure name as part of the error message? For example, the default behavior of the MESSAGE facility is to always include the procedure name along with an error message.

On Oct 12, 3:45 pm, Chris Torrence <gorth...@gmail.com> wrote:

> workings of the library. If you are writing your own routines, or are
> debugging an existing routine, then I would recommend that you disable
> the on_error,2 command until you have completed your routine and are
> ready to "release" it.

One minor problem with this is that it is not easy to disable ON_ERROR, 2 commands. I am currently upgrading a well-established library of ~50 routines, so I would need to first comment out all the ON_ERROR statements, install the upgrades, and then uncomment the ON_ERROR lines.
Besides Paul van Delst's warning of the need to edit "fixed" code, it is all a bit of a pain.

I always wished there was a global way to change the ON_ERROR values. I could do this now by creating a new system variable, say !ON_ERROR, and starting all my code with On_ERROR, !ON_ERROR. Then if I wanted to go into debug mode, I would set !ON_ERROR = 3. But there are other problems with introducing a new system variable. --
Wayne

Subject: Re: IDL 8.0 bug -- line number of errors not given

On Oct 12, 11:46 pm, David Fanning <n...@dfanning.com> wrote:
> Paulo Penteado writes:
>> I was wondering if this was the case, since I agree this was
>> desirable.
>
> Humm. In what sense of the word do you find this "desirable"?

The way I see it, debugging when developing the code is usually most helped by the default on_error (0), so that one is left at the scope and line of the error, and can navigate between scopes to inspect things.

Using 2 for on_error could be appropriate only when one expects that the only source of problems occurring in the routine is in the parameters it gets passed, and every error inside the routine is being handled. So it would just get back to the caller, using message to inform the caller what was wrong with the parameters. In that situation, it would be less confusing for the user to know that the problem was with the parameters used in the routine call (through the message passed), instead of seeing a long sequence of errors from some routine called several levels below, which may not make it obvious why that routine call created a problem. In that sense, this is locating the source of the error more precisely, as it was the parameters used in the routine call, not some obscure code deep in some other routine, where the error was thrown.

The trouble with 2 for on_error is when the error is unhandled. Then just getting some like "% Specified offset to array is out of range: A" is really unhelpful. In such a situation, I would argue that 'on_error,0' should have been used.

>
>> Not showing the full trace is consistent with returning to the caller.
>
> Returning to the caller for what purpose?

For the purpose of saying (through message) that the problem was the way the routine was called, not leaving some unhandled error to happen inside the routine.

>
>> That is, the code below behaves as I expected, and agree that should
>> be the case:
>
> Why is this what you expected? Because of your experience
> with other software? What other software acts like this?

> You thought IDL's behavior in the past was inconsistent with
> good programming practices?

When all errors get handled in the routine, it can use 'on_error,2' to expose only an informative error message telling why the parameters used on its call caused trouble. What is publicly exposed is only the routine's interface and its error messages, not unhandled errors that are only informative if one knows the implementation.

I just thought that showing the full trace when a simple informative error message is given and scope is returned to the caller makes things a little more confusing. If one wants to see all the details, it is better to remain in the scope (and line) of the error, not return to the caller. In a dynamic language like IDL, just statically (out of the scope) looking at the line where the error was thrown may not say much anyway: it would be necessary to still be in scope, to inspect the variables used in that line, to figure out what happened. In such a situation, the traceback would help most to know where to add a breakpoint, to run the program again and stop in the line of trouble. That is, to get the same as if on_error was 0 and execution stopped at the error.

Returning to the caller is more consistent with just saying that the caller is at fault (and why) instead of showing what was going on inside.

Java has a much more elaborate error handling structure, and its checked exceptions, declared with throws, seem to me a similar idea, of a routine telling that it may give some error back to the caller (supposedly with an informative message), to let one know how what the problem was with the routine call.

Or, to put it another way, add an 'on_error,2' to a routine is somewhat like turning off traceback on a compiled language, to rely solely on the program's messages to inform of problems.

>
>> function test1
>> On_error,2
>> a = bindgen(32)
>> c = long(a,30,1)
>> return,1
>> end
>
>> pro test
>> print,test1()
>> return
>> end

>
>> IDL> test
>> % Compiled module: TEST.
>> % Specified offset to array is out of range: A.
>> % Execution halted at: TEST 9 /home/penteado/idl/
>> test.pro
>> % $MAIN$
>
>> As the trace shows where the place where the routine with 'on_error,2'
>> was called.
>
> Well, I agree this is the place it was called. But what possible
> good is this information? How would you use this information?
>
> Why would you use ON_ERROR, 2? Should it be eliminated from
> the language? How would you use it now that it is working
> as you expect it to?

In this case it is no good because the error was not handled, there is no message to inform of the problem. In this case I think it would be better to leave 0 for on_error.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Wed, 13 Oct 2010 04:55:55 GMT
View Forum Message <> Reply to Message

On Oct 12, 11:42 pm, wlandsman <wlands...@gmail.com> wrote:
> But if I use CATCH I need to add a whole paragraph somewhere (where?)
> in the code
>
> Catch, theError
> IF theError NE 0 THEN BEGIN
> Catch, /Cancel
> void = Error_Message()
> RETURN
> END
>
> So what are the advantages of using CATCH? (This is not a trick
> question -- I've had a mental block about how to use CATCH.) --Wayne

To just throw an error message and return, not many advantages indeed: One is having only one place to send a common error message, instead of several tests, all with the same argument to message. Another is not stopping execution, like David mentioned.

But I find that the main use of catch is to handle the errors, not just send error messages. It can be part of the approach of 'asking

for forgiveness' (handling the errors) instead of 'asking for
permission' (testing to avoid errors). Sometimes it is much easier to
rely on an error occurring (and handling it) than to test for many
potentially complicated possibilities.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 12:43:32 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> The way I see it, debugging when developing the code is usually most
> helped by the default on_error (0), so that one is left at the scope
> and line of the error, and can navigate between scopes to inspect
> things.
>
> Using 2 for on_error could be appropriate only when one expects that
> the only source of problems occurring in the routine is in the
> parameters it gets passed, and every error inside the routine is being
> handled. So it would just get back to the caller, using message to
> inform the caller what was wrong with the parameters. In that
> situation, it would be less confusing for the user to know that the
> problem was with the parameters used in the routine call (through the
> message passed), instead of seeing a long sequence of errors from some
> routine called several levels below, which may not make it obvious why
> that routine call created a problem. In that sense, this is locating
> the source of the error more precisely, as it was the parameters used
> in the routine call, not some obscure code deep in some other routine,
> where the error was thrown.
>
> The trouble with 2 for on_error is when the error is unhandled. Then
> just getting some like "% Specified offset to array is out of range:
> A" is really unhelpful. In such a situation, I would argue that
> 'on_error,0' should have been used.

OK, thank you, Paulo. I think I understand this argument
in principal now, anyway. My biggest problem with it is that
I don't think this is how most people write code. I know its
not how I write code. I typically use On_Error, 2 in program
utility routines. I want *any* error in these utility routines
to be returned to the main program module, where I am catching
and reporting errors.

It's all well and good to say its clear that the
"general philosophy" of error handling is such and so.
But when you turn sometimes poorly documented software
over to real users, they use what you give them, and often

in ways you never anticipated. I think it is unfair (not to say arrogant) to suddenly (after 20 years!) change the rules and then claim that the use of "library" in the documentation makes it clear what the "general philosophy" was all along. Hell, I barely understand it when it's spelled out for me!

I just think that if one of your goals is backward compatibility (and I am *extremely* appreciative that this is one of ITTVIS's goals), then you don't change something as fundamental as your error handling mechanism, no matter how poorly thought out you think it might have been originally from a prospective 20 years on.

I have no problems with tweaking it to do something else, but you can't fundamentally change the way it works just because you suddenly "discovered" in your own programs that it throws long error messages. Yes, it does. And there is a lot of code out there that relies on just this fact.

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: IDL 8.0 bug -- line number of errors not given
Posted by Paul Van Delst[1] on Wed, 13 Oct 2010 14:50:37 GMT
View Forum Message <> Reply to Message

wlandsman wrote:
> On Oct 12, 6:16 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
>
>> Maybe people should start switching to using CATCH instead for their day-to-day error handling?
>
> OK, here's my simpleton reason for not using CATCH. If I use
> ON_ERROR I just add 1 line of code at the beginning of each procedure
>
> On_error, 2
>

> But if I use CATCH I need to add a whole paragraph somewhere (where?)
> in the code
>
> Catch, theError
> IF theError NE 0 THEN BEGIN
> Catch, /Cancel
> void = Error_Message()
> RETURN
> END

Oh yes, I agree - using CATCH is a more verbose method. Applying the DRY[*] principle, what I do for a particular
application is to create include files for functions and procedures (say "func_err_handler.pro" and "pro_err_handler.pro") and then simply include them right at the top of functions/procedures. E.g:

```
function testfunc, a, b, c
  @func_err_handler
  ....
end

pro testpro, a, b, c
  @pro_err_handler
  ....
end
```

I don't think it's an ideal solution (it's still a bit "wet") but I have isolated everything CATCH-y in two files. Easy
to change if I need to.

With a bit more thought, one could probably come up with content for these include files that are applicable throughout
ones entire library of routines.

> So what are the advantages of using CATCH?

It's not "ON_ERROR, 2"? :o)

> (This is not a trick
> question -- I've had a mental block about how to use CATCH.) --Wayne

cheers,

paulv

[*] http://c2.com/cgi/wiki?DontRepeatYourself

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 15:03:38 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> With a bit more thought, one could probably come up with content for these include files that are applicable throughout
> ones entire library of routines.

Easier in theory than in practice, I think. Naturally, there are plenty of places to use "generic" Catch handlers. (Widget event handlers come immediately to mind.)

But I haven't had much luck modifying generic handlers to be "more useful". I usually find I need to customize each for the particular need at the time. I guess that's why I just type the paragraph out.

Cheers,

David

P.S. Let's just say my fingers had no luck at all learning EMACS, but they can type Catch paragraphs in a hurry! :-)

--
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: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Wed, 13 Oct 2010 16:46:54 GMT
View Forum Message <> Reply to Message

It seems that I just got bitten by this new behavior. A simple example that shows the problem:

```
IDL> w=window()
IDL> w.title='a'
% Expression must be a structure in this context: OTITLE.
% Execution halted at: $MAIN$
```

So it seems that somewhere in the Graphics routines (I am guessing in
a setproperty method) there is a bug, but the error message does not
tell where. I will have to start digging now.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Wed, 13 Oct 2010 17:06:36 GMT
View Forum Message <> Reply to Message

On Oct 13, 1:46 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> It seems that I just got bitten by this new behavior. A simple example
> that shows the problem:
>
> IDL> w=window()
> IDL> w.title='a'
> % Expression must be a structure in this context: OTITLE.
> % Execution halted at: $MAIN$
>
> So it seems that somewhere in the Graphics routines (I am guessing in
> a setproperty method) there is a bug, but the error message does not
> tell where. I will have to start digging now.

And this confirmed the existence of another problem, which I had been
suspecting existed: when I set breakpoints in programs in IDL's
library, they get ignored. It is not an issue of not putting them in
the right routine by changing the source code (in this particular
instance, of graphicswin__define.pro). And when I move that file from
the IDL library directory to my own, the breakpoints work. Is this
intended? Is there a way around it (besides moving the entire library
directory somewhere else)?

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by chris_torrence@NOSPAM on Wed, 13 Oct 2010 17:47:01 GMT
View Forum Message <> Reply to Message

On Oct 13, 11:06 am, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Oct 13, 1:46 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
>
>> It seems that I just got bitten by this new behavior. A simple example
>> that shows the problem:
>
>> IDL> w=window()
>> IDL> w.title='a'
>> % Expression must be a structure in this context: OTITLE.
>> % Execution halted at: $MAIN$

>
>> So it seems that somewhere in the Graphics routines (I am guessing in
>> a setproperty method) there is a bug, but the error message does not
>> tell where. I will have to start digging now.
>
> And this confirmed the existence of another problem, which I had been
> suspecting existed: when I set breakpoints in programs in IDL's
> library, they get ignored. It is not an issue of not putting them in
> the right routine by changing the source code (in this particular
> instance, of graphicswin__define.pro). And when I move that file from
> the IDL library directory to my own, the breakpoints work. Is this
> intended? Is there a way around it (besides moving the entire library
> directory somewhere else)?

Hi Paulo,

A few answers:

1. That w.title is a bug that is fixed in the upcoming patch release.

2. You might want to use the undocumented /DEBUG keyword when using the new graphics. This should cause execution to halt where the error occurs (it disables the on_error and catch behavior).

3. Regarding the breakpoints - this sounds like a bug - you should be able to set breakpoints in the lib directory. You might try setting the breakpoint, and then hitting the compile button in the Workbench.

4. You might find it useful to create a project in the Workbench that points to the lib directory. Just do "New IDL Project", then choose "Create the new project from an existing directory", and choose the Lib directory. Be sure to turn off the "Update IDL path" since you'll already have the Lib on your path.

Once you have the Lib directory in a project, you can do things like Ctrl+H to search across all files, Ctrl+Shift+R to open up a particular file, etc.

-Chris
ITTVIS

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 18:33:33 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> And this confirmed the existence of another problem, which I had been
> suspecting existed: when I set breakpoints in programs in IDL's
> library, they get ignored.

Too funny! :-)

Let us know how this search goes. I've noticed
the breakpoint problem, too.

Cheers,

David

P.S. Do you ever get the feeling that mortals are
not *meant* to be debugging ITTVIS supplied code. :-)

--
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: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 18:35:34 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> It seems that I just got bitten by this new behavior. A simple example
> that shows the problem:
>
> IDL> w=window()
> IDL> w.title='a'
> % Expression must be a structure in this context: OTITLE.
> % Execution halted at: $MAIN$
>
> So it seems that somewhere in the Graphics routines (I am guessing in
> a setproperty method) there is a bug, but the error message does not
> tell where. I will have to start digging now.

Yes, try grepping all your ON_ERROR,2 lines and changing
them to ON_ERROR,0. There should only be a couple of thousand. ;-)

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: IDL 8.0 bug -- line number of errors not given
Posted by Paul Van Delst[1] on Wed, 13 Oct 2010 20:53:38 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Paulo Penteado writes:
>
>> It seems that I just got bitten by this new behavior. A simple example
>> that shows the problem:
>>
>> IDL> w=window()
>> IDL> w.title='a'
>> % Expression must be a structure in this context: OTITLE.
>> % Execution halted at: $MAIN$
>>
>> So it seems that somewhere in the Graphics routines (I am guessing in
>> a setproperty method) there is a bug, but the error message does not
>> tell where. I will have to start digging now.
>
> Yes, try grepping all your ON_ERROR,2 lines and changing
> them to ON_ERROR,0. There should only be a couple of thousand. ;-)

This is actually quite trivial.

I had to do this a few years ago on my Fortran codes - funnily enough it was to do with the error handling module!. I
needed (wanted?) to change all instances of "error_handler" to "Message_Handler".

Suffice it to say over the decades I had a fair amount of Fortran code that used my error_handler module. The initial
solution was to use ruby on the command line:

$ ruby -pi.bak -e "gsub(/error_handler/i,'Message_Handler')" *.f90

Bugger me if that didn't work. I thought it was neato enough to create a generic ruby script to do similar stuff (shown
below).

Let me know if it works on your code..... :o)

cheers,

paulv

p.s. A bloke I work with did exactly the same thing but using perl.

<-----begin----->
```
#!/usr/bin/env ruby

# == Synopsis
#
# Substitute strings in files
#
# == Usage
#
# strsub.rb [OPTION] old new file1 [file2 file3 file4 ...]
#
# == Options
#
# --help  (-h):
#    you're looking at it
#
# --ignore-case  (-i)
#    ignore the case of the string to replace
#
# --no-backup  (-n):
#    do not create a backup file before modification.
#    Backup file name is "filename".bak
#
# == Arguments
#
# old:
#    string to replace
#
# new:
#    replacement string
#
# file1 [file2 file3 file4 ...]:
#    files on which to perform the substitution
#
#
# Written by:: Paul van Delst, 11-Aug-2006
#
```

```
require 'getoptlong'
require 'rdoc/usage'

# Specify accepted options
options=GetoptLong.new(
  [ "--help",       "-h", GetoptLong::NO_ARGUMENT ],
  [ "--ignore-case","-i", GetoptLong::NO_ARGUMENT ],
  [ "--no-backup",  "-n", GetoptLong::NO_ARGUMENT ] )

# Define backup default
$-i=".bak"

# Define defaults
ignorecase=0

# Parse the command line options
begin
  options.each do |opt, arg|
    case opt
      when "--help"
        RDoc::usage
        exit 0
      when "--ignore-case"
        ignorecase=Regexp::IGNORECASE
      when "--no-backup"
        $-i=""
    end
  end
rescue StandardError=>errormessage
  puts "ERROR: #{error_message}"
  RDoc::usage
  exit 1
end

# Check for args
if ARGV.empty?
  puts "ERROR: No arguments supplied"
  RDoc::usage
  exit 1
end

# Get the strings
old=ARGV.shift
new=ARGV.shift

# Build the regex
re=Regexp.new(/#{old}/,ignorecase)
```

```
# Inplace edit the file
ARGF.each do |line|
  line.gsub!(re, new)
  puts line
end
<-----end----->
```

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Wed, 13 Oct 2010 21:08:19 GMT
View Forum Message <> Reply to Message

On Oct 13, 2:47 pm, Chris Torrence <gorth...@gmail.com> wrote:
> A few answers:
>
> 1. That w.title is a bug that is fixed in the upcoming patch release.

OK. Is there an estimate of when that release will come out?

For now I will either use text(), or (more likely) resort to directly
getting the title object and changing its string. I encountered that
problem while making a setproperty method for a class to provide
multiplot-like functionality for the new graphics (http://
 groups.google.com/group/comp.lang.idl-pvwave/browse_thread/t hread/
d51b661feb93451f/).

>
> 2. You might want to use the undocumented /DEBUG keyword when using
> the new graphics. This should cause execution to halt where the error
> occurs (it disables the on_error and catch behavior).

That seemed to have no effect. At least for the methods I was looking
at (setproperty in graphicswin and graphic).

> 3. Regarding the breakpoints - this sounds like a bug - you should be
> able to set breakpoints in the lib directory. You might try setting
> the breakpoint, and then hitting the compile button in the Workbench.

Interesting. That makes the breakpoint work. I always recompiled after
setting a breakpoint, as I am aware that it is sometimes (always?)
necessary. But I usually recompile in the (Workbench's) console (I am
used to frequently doing a .full_reset_session, to make sure that
everything will be recompiled).

So the breakpoint is ignored with autocompilation or compilation with
a .compile, but not with the compile button. And that behavior is
different if the routine is in IDL's library. I checked, running the
Workbench as root, if it might be due to an inability to write to the

routine's directory, and found that it made no difference.

> 4. You might find it useful to create a project in the Workbench that
> points to the lib directory. Just do "New IDL Project", then choose
> "Create the new project from an existing directory", and choose the
> Lib directory. Be sure to turn off the "Update IDL path" since you'll
> already have the Lib on your path.
>
> Once you have the Lib directory in a project, you can do things like
> Ctrl+H to search across all files, Ctrl+Shift+R to open up a
> particular file, etc.

That does not normally work, as a project requires write permission to the directory, which I only provide temporarily, when testing things like this.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by penteado on Wed, 13 Oct 2010 21:10:47 GMT
View Forum Message <> Reply to Message

On Oct 13, 5:53 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
> Let me know if it works on your code..... :o)
>
> cheers,
>
> paulv
>
> p.s. A bloke I work with did exactly the same thing but using perl.

I was considering a combination of xargs and sed. But if the debug keyword can work, that would not be necessary.

Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 21:33:33 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> The initial
> solution was to use ruby on the command line:
>
> $ ruby -pi.bak -e "gsub(/error_handler/i,'Message_Handler')" *.f90
>
> Bugger me if that didn't work. I thought it was neato enough to create a generic ruby script to do

similar stuff (shown
> below).
>
> Let me know if it works on your code..... :o)
>
> p.s. A bloke I work with did exactly the same thing but using perl.

I'll add this to my chapter on using free software
to get IDL to do what you want it to do. ;-)

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: IDL 8.0 bug -- line number of errors not given
Posted by Paul Van Delst[1] on Wed, 13 Oct 2010 22:07:20 GMT
View Forum Message <> Reply to Message

Paulo Penteado wrote:
> On Oct 13, 5:53 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
>> Let me know if it works on your code..... :o)
>>
>> cheers,
>>
>> paulv
>>
>> p.s. A bloke I work with did exactly the same thing but using perl.
>
> I was considering a combination of xargs and sed. But if the debug
> keyword can work, that would not be necessary.

True - it would be a nice convention for debug keywords to exist in IDL routines in general... in some fashion. I
actually stick in an (also undocumented!) debug keyword in all my new IDL routines that control the error handling, i.e.
force any errors to cause execution to halt in the routine.

Regarding the ON_ERROR issue, it would be more difficult to do than a simple string substitution, but one could figure
out a way to modify their routines that do something like

```
  PRO mywhatsit, arg1, arg2, etc...
    ON_ERROR, 2
```

to

```
  PRO mywhatsit, arg1, arg2, etc..., Debug=Debug
    ON_ERROR, KEYWORD_SET(Debug) ? 0 : 2
```

Problem solved! :o)

cheers,

paulv
