Subject: Re: main or procedure
Posted by David Fanning on Wed, 20 Jun 2007 12:20:37 GMT
View Forum Message <> Reply to Message

sujian@gmail.com writes:

> When I .compile a procedure, it says,
> You compiled a main program while inside a procedure. Returning.
>
> And all the information before the compilation was lost.
>
> How to fix it?

You need to do two things: (1) Add more error handling
to your code, so that when a program crashes it goes
back to the main program level (ON_ERROR, 1), and (2)
learn to type RETALL *immediately* when your program
crashes, and certainly before you do any more compiling.

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: main or procedure
Posted by Conor on Wed, 20 Jun 2007 12:39:14 GMT
View Forum Message <> Reply to Message

On Jun 20, 8:20 am, David Fanning <n...@dfanning.com> wrote:
> suj...@gmail.com writes:
>> When I .compile a procedure, it says,
>> You compiled a main program while inside a procedure. Returning.
>
>> And all the information before the compilation was lost.
>
>> How to fix it?
>
> You need to do two things: (1) Add more error handling
> to your code, so that when a program crashes it goes
> back to the main program level (ON_ERROR, 1), and (2)
> learn to type RETALL *immediately* when your program
> crashes, and certainly before you do any more compiling.
>

> 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.")

What's happening is the effects of variable scope. IDL has various levels of "scope" for every variable it creates. When you initially launch IDL, you are on the "MAIN" level. Everytime you run a program or function, that program runs in its own separate scope, that is below the "level" of the main level. Most importantly, it is entirely separate from the main level with an independent set of variables. So, if you have a variable named 'asdf' in your main level, and then you launch a program, the program in question won't be able to access that same 'asdf' variable, because your program runs in a different scope. By default, the variables used in a program can't be used in the main level, and vice-versa. Now, when a program encounters an error and quits it returns you to the idl prompt, but it returns you at the level of the program that had the error, not at the main level. What this means is that when your program encounters an error you won't immediately have access to the variables you declared initially in the main level. In your case, another important point is that when IDL is inside the scope of a program that had an error, and that program is re-compiled, then IDL automatically returns you to the main level and you automatically lose any variables that had existed in the program's scope. That's what is happening to you. At some point you are running a routine that gives an error. Then, you are continuing to use the IDL command line, creating variables and doing whatever it is you do. When you re-compile the program that had the original error, IDL automatically returns you to the main level, and all those variables you created go out of scope and are deleted. The solution is quite simple. Just do what David pointed out. If you set the ON_ERROR command inside your program like he noted, then idl will return you to the main level when the program encounters an error, rather than to the program level. This isn't the best solution during development though, since returning to the main level means you lose all your program variables and therefore you can't examine the state of your program to see where the problem was. The other solution is to simply type 'retall' before you resume working with the IDL command line. 'retall' will return you to the main programming level, once again giving you access to any variables you had created previously.

Subject: Re: main or procedure

Posted by Carsten Lechte on Wed, 20 Jun 2007 13:07:44 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> You need to do two things: [...]

Yes, but that only serves to destroy his context more quickly.
The way I understood the question is, how can one debug a
procedure way down the call chain and still compile some
helper code while doing this.

IDLWAVE has this nice feature idlwave-shell-run-region, but
this compiles a new MAIN program, so you cannot use it while
debugging a running program. I find this annoying to no end.

So, the workaround is to avoid compiling new main programs
while debugging.

chl

Subject: Re: main or procedure
Posted by David Fanning on Wed, 20 Jun 2007 14:00:25 GMT
View Forum Message <> Reply to Message

Carsten Lechte writes:

> So, the workaround is to avoid compiling new main programs
> while debugging.

If programs are written correctly, they don't have to
be compiled manually, hence they work on ANY level. :-)

http://www.dfanning.com/tips/namefiles.html

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: main or procedure

Posted by Carsten Lechte on Wed, 20 Jun 2007 14:24:56 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> If programs are written correctly, they don't have to
> be compiled manually, hence they work on ANY level. :-)

If programs were written correctly, there would be no need to debug them in the first place. Since they aren't, there is a need, hence the trap of writing some ad-hoc data displaying code to find out why the program stopped, and running said code by compiling it into a new main program (which is what IDLWAVE's very useful idlwave-shell-run-region function will do).

Then realising you should have run the code by either entering it into the command line, or putting it into a named function or procedure. Then swear profusely at IDL for returning to the main level and destroying all the data you would have needed for proper error diagnostic.

Of all the crimes against usability, throwing away the user's data without asking is one of the worst.

chl

Subject: Re: main or procedure
Posted by Marshall Perrin on Wed, 20 Jun 2007 14:39:14 GMT
View Forum Message <> Reply to Message

David Fanning <news@dfanning.com> wrote:
> sujian@gmail.com writes:
>
>> When I .compile a procedure, it says,
>> You compiled a main program while inside a procedure. Returning.
>>
>> And all the information before the compilation was lost.
>>
>> How to fix it?
>
> You need to do two things: (1) Add more error handling
> to your code, so that when a program crashes it goes
> back to the main program level (ON_ERROR, 1), and (2)
> learn to type RETALL *immediately* when your program
> crashes, and certainly before you do any more compiling.

It feels odd to disagree with the great Dr. Fanning, but in this case

I want to offer a counter-suggestion to the above advice. I much prefer for my programs to stop at the point of failure when they crash, because that way I have a chance to examine the variables and program state and try to figure out what went wrong. I *hate* it when programs return back to the main level, since then it's usually much harder to understand the situation that caused the crash.

Besides, often when the error is in a subroutine, it can be fixed and the program execution resumed from that point. This can be a big timesaver when running some computationally intensive code that you don't want to have to restart from the very beginning:

```
dxfinish                ; set breakpoint after broken subroutine
return                  ; or return,0 if you're in a function
.comp fixed_subroutine.pro   ; after you've fixed the error
fixed_subroutine, /arguments ; paste in the relevant line to re-run the
      ; routine, which will hopefully now work
```

I think the difference between David's approach and mine probably comes down to how we structure our main programs. His are probably mostly Widget programs, so if you retall to the main level, hopefully all your state is retained and you can resume your work without losing all your information. I on the other hand usually just run long computations or data analysis routines in regular procedures, so a retall in the middle of that would indeed lose all my state. Hence my deep aversion to "on_error, 1". I'd argue that these are equally good, if different, approaches, but maybe other folks here will care to chime in with their opinions.

For the original poster: The reason it returned automatically to the top level in your case is that you compiles a main routine (i.e. a file with IDL code but no "PRO whatever" or "FUNCTION whatever") while inside a procedure or function. That, you can't do. You can compile a PRO or FUNCTION while inside another function (but not while inside itself), but main-level routines can only be compiles from the main level. Which means if you want to have a batch of commands that you can execute interactively from the command line while at a breakpoint, you either need to package it up inside a procedure, or you need to run it with the @FILENAME syntax (with all the hassles about loops and punctuation that entails.)

- Marshall

Subject: Re: main or procedure
Posted by David Fanning on Wed, 20 Jun 2007 14:40:29 GMT
View Forum Message <> Reply to Message

Carsten Lechte writes:

> Then realising you should have run the code by either entering
> it into the command line, or putting it into a named function
> or procedure. Then swear profusely at IDL for returning to the
> main level and destroying all the data you would have needed
> for proper error diagnostic.

Honestly, Carsten, I think you are swearing at the wrong
thing here. As far as I can tell, there is nothing about
IDL that is causing you any problems.

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: main or procedure
Posted by David Fanning on Wed, 20 Jun 2007 14:56:14 GMT
View Forum Message <> Reply to Message

Marshall Perrin writes:

> It feels odd to disagree with the great Dr. Fanning, but in this case
> I want to offer a counter-suggestion to the above advice.

Actually, we probably don't disagree. I've pretty much
given up EVER using ON_ERROR,1. If I need error handling,
I typically use a CATCH.

My impression was that the person asking the question
was not very knowledgeable about IDL. I've learned it is
better to start with the simple answers, rather than laying
the whole sordid picture out all at once. I guess we could ask
him. Sukye, did you understand ANYTHING Carsten Lechte said?

> For the original poster: The reason it returned automatically to the
> top level in your case is that you compiles a main routine (i.e.
> a file with IDL code but no "PRO whatever" or "FUNCTION whatever")
> while inside a procedure or function. That, you can't do.

I don't think this is what happened, but perhaps it is the
message that is confusing. This is the message
you get if you crash in any IDL procedure or function
and are stopped somewhere other than at the main level
and you try to compile the same procedure or function that
failed. I'd say this is 95% certain what has happened here.
A simple RETALL would eliminate the message, but, of course,
all variables created at that level would be lost. The good
news, of course, is that you will re-discover all those
variables you ACTUALLY created at the main level. :-)

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: main or procedure
Posted by Conor on Wed, 20 Jun 2007 15:42:09 GMT
View Forum Message <> Reply to Message

On Jun 20, 10:56 am, David Fanning <n...@dfanning.com> wrote:
> Marshall Perrin writes:
>> It feels odd to disagree with the great Dr. Fanning, but in this case
>> I want to offer a counter-suggestion to the above advice.
>
> Actually, we probably don't disagree. I've pretty much
> given up EVER using ON_ERROR,1. If I need error handling,
> I typically use a CATCH.
>
> My impression was that the person asking the question
> was not very knowledgeable about IDL. I've learned it is
> better to start with the simple answers, rather than laying
> the whole sordid picture out all at once. I guess we could ask
> him. Sukye, did you understand ANYTHING Carsten Lechte said?
>
>> For the original poster: The reason it returned automatically to the
>> top level in your case is that you compiles a main routine (i.e.
>> a file with IDL code but no "PRO whatever" or "FUNCTION whatever")
>> while inside a procedure or function. That, you can't do.
>
> I don't think this is what happened, but perhaps it is the
> message that is confusing. This is the message
> you get if you crash in any IDL procedure or function

> and are stopped somewhere other than at the main level
> and you try to compile the same procedure or function that
> failed. I'd say this is 95% certain what has happened here.
> A simple RETALL would eliminate the message, but, of course,
> all variables created at that level would be lost. The good
> news, of course, is that you will re-discover all those
> variables you ACTUALLY created at the main level. :-)
>
> 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.")

That's what I assumed happened as well. When I first started using IDL I had the exact opposite problem. I'd create all my variables and what-not on the main level, and then my program would crash and all my variables would dissappear. I had no idea what was going on, and no one else I asked seemed to know either. It really irked me because I was working with some large text files, and loading the data into IDL was a particularly lengthy process (mainly because I had written a horribly inefficient program that read the files into IDL). Every time my program crashed I had to wait another 5 minutes for my data to reload. Then, once it was done reloading, I would fix the bug and recompile the program, bringing myself back to the main level and restoring all the data I thought I had "lost" while I was in the program scope. Of course I had already re-loaded the data, so I didn't notice my missing data magically re-appearing when I re-compiled my program, and I didn't know what the "procedure compiled while active: returning' message meant, so I never realized what was happening. I continued in this fashion for a couple weeks before I finally figured out what was going on :)

Subject: Re: main or procedure
Posted by Carsten Lechte on Wed, 20 Jun 2007 16:03:30 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> As far as I can tell, there is nothing about
> IDL that is causing you any problems.

Well, I have outlined my problem, and I get bitten by it again and again. It is caused by IDL being IDL and myself relying a lot on on-the-fly compilation

of small code fragments. Other people will have
different styles (and different objectives) and IDL's
creators cannot forsee every way their software might
be used.

It is really a problem of how the human mind works:
The situations where my ingrained working style causes
me data loss are so rare that I do not become aware of
them until about one millisecond after it is too late.
This is a prime example of where the software should help
the user avoid this kind of error.

So, you other IDLWAVE users: have you eschewed the command
line and become addicted to C-c C-d C-e, like I have?

chl

Subject: Re: main or procedure
Posted by Carsten Lechte on Wed, 20 Jun 2007 16:12:41 GMT
View Forum Message <> Reply to Message

Conor wrote:
> Then, once it was done reloading, I would fix the bug and
> recompile the program, bringing myself back to the main level and
> restoring all the data I thought I had "lost" while I was in the
> program scope.

;-) Bonus points if you loaded different data and now wonder
how IDL still knows about the old. Yes, it happened to me.

chl

Subject: Re: main or procedure
Posted by Vince Hradil on Wed, 20 Jun 2007 16:47:58 GMT
View Forum Message <> Reply to Message

On Jun 20, 11:12 am, Carsten Lechte <c...@toppoint.de> wrote:
> Conor wrote:
>> Then, once it was done reloading, I would fix the bug and
>> recompile the program, bringing myself back to the main level and
>> restoring all the data I thought I had "lost" while I was in the
>> program scope.
>

> ;-) Bonus points if you loaded different data and now wonder
> how IDL still knows about the old. Yes, it happened to me.
>
> chl

To be honest I'm having trouble following this thread. It all has to do with variable scope and keeping track of "where you are", so to speak. Not an IDL problem, but clearly EBKAC!

Subject: Re: main or procedure
Posted by Paul Van Delst[1] on Wed, 20 Jun 2007 17:48:27 GMT
View Forum Message <> Reply to Message

Carsten Lechte wrote:
> David Fanning wrote:
>> As far as I can tell, there is nothing about IDL that is causing you
>> any problems.
>
> Well, I have outlined my problem, and I get bitten
> by it again and again. It is caused by IDL being IDL
> and myself relying a lot on on-the-fly compilation
> of small code fragments. Other people will have
> different styles (and different objectives) and IDL's
> creators cannot forsee every way their software might
> be used.
>
> It is really a problem of how the human mind works:
> The situations where my ingrained working style causes
> me data loss are so rare that I do not become aware of
> them until about one millisecond after it is too late.

Please don't take this the wrong way, but this reminds me of the old chestnut:

Patient: Doctor, it hurts when I do this.

Doctor: Well, don't do that.

:o)

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx

Subject: Re: main or procedure
Posted by Carsten Lechte on Wed, 20 Jun 2007 18:50:38 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
> Patient: Doctor, it hurts when I do this.
>
> Doctor: Well, don't do that.

Hey, that sounds so fatalistic. I'll start hacking into
IDLWAVE to solve my little problem right away!

chl

Subject: Re: main or procedure
Posted by Conor on Wed, 20 Jun 2007 18:54:47 GMT
View Forum Message <> Reply to Message

On Jun 20, 12:12 pm, Carsten Lechte <c...@toppoint.de> wrote:
> Conor wrote:
>> Then, once it was done reloading, I would fix the bug and
>> recompile the program, bringing myself back to the main level and
>> restoring all the data I thought I had "lost" while I was in the
>> program scope.
>
> ;-) Bonus points if you loaded different data and now wonder
> how IDL still knows about the old. Yes, it happened to me.
>
> chl

lol! Now that would be frustrating.

Subject: Re: main or procedure
Posted by David Fanning on Wed, 20 Jun 2007 19:14:29 GMT
View Forum Message <> Reply to Message

Carsten Lechte writes:

> Hey, that sounds so fatalistic. I'll start hacking into
> IDLWAVE to solve my little problem right away!

Perhaps you are unaware that the only connection between
IDL and IDLWAVE is that the makers of the former gave
the programmers of the latter a T-shirt in about 1993.
I still think you might be looking down the wrong end

of the gun, at least if my suspicion that there is more
than a little bit of sarcasm in the statement above is
true. :-)

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: main or procedure
Posted by Carsten Lechte on Wed, 20 Jun 2007 20:13:45 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> I still think you might be looking down the wrong end
> of the gun, at least if my suspicion that there is more
> than a little bit of sarcasm in the statement above is
> true. :-)

No sarcasm intended, just cheerful optimism. Since I cannot
change the IDL sourcecode to suit my whims, I will have to
adapt IDLWAVE, the mediating layer between me and IDL, to
keep me from inadvertently compiling a main program when I
am debugging. That's the price I pay for my what I now know
to be idiosyncratic use of IDL.

chl

Subject: Re: main or procedure
Posted by JD Smith on Wed, 20 Jun 2007 23:36:30 GMT
View Forum Message <> Reply to Message

On Wed, 20 Jun 2007 09:47:58 -0700, hradilv wrote:

> On Jun 20, 11:12 am, Carsten Lechte <c...@toppoint.de> wrote:
>> Conor wrote:
>>> Then, once it was done reloading, I would fix the bug and
>>> recompile the program, bringing myself back to the main level and
>>> restoring all the data I thought I had "lost" while I was in the
>>> program scope.

>>
>> ;-) Bonus points if you loaded different data and now wonder
>> how IDL still knows about the old. Yes, it happened to me.
>>
>> chl
>
> To be honest I'm having trouble following this thread. It all has to
> do with variable scope and keeping track of "where you are", so to
> speak. Not an IDL problem, but clearly EBKAC!

Allow me to step in with a plug for IDLWAVE. If you are running in the IDLWAVE shell, and a program stops in the middle, for whatever reason (STOP, a breakpoint, a run-time error, C-c, etc.), the status bar in the buffer will show you exactly "where" you are in the calling stack, e.g.

[0:MYROUTINE]

If you navigate the call stack, this will update as well. If you're at the main level, the stack status will clear.

JD

Subject: Re: main or procedure
Posted by JD Smith on Thu, 21 Jun 2007 00:02:14 GMT
View Forum Message <> Reply to Message

On Wed, 20 Jun 2007 22:13:45 +0200, Carsten Lechte wrote:

> David Fanning wrote:
>> I still think you might be looking down the wrong end
>> of the gun, at least if my suspicion that there is more
>> than a little bit of sarcasm in the statement above is
>> true. :-)
>
> No sarcasm intended, just cheerful optimism. Since I cannot
> change the IDL sourcecode to suit my whims, I will have to
> adapt IDLWAVE, the mediating layer between me and IDL, to
> keep me from inadvertently compiling a main program when I
> am debugging. That's the price I pay for my what I now know
> to be idiosyncratic use of IDL.

That's not a bad idea... probably C-c C-d C-e should be disabled when stopped in a program. Of course, if you "retall" yourself at the command line (instead of using IDLWAVE to do that for you), it can get confused and think you are still stopped.

I tried this and it works well, prompting if you are stopped at a breakpoint, error, etc. Look for it in the next release, which is looking more and more like it will be 6.2 (some fixes since the Emacs 22 v6.1).

You will notice I made the shortcut "e" in electric debug mode do something else beside run-region (eval an expression). If your block of code would run line by line, you can always drop the whole thing onto the command line.

I use run-region a reasonable amount, but never when debugging. Be sure to try all the rich examine functionality, to print or otherwise inspect variables or expressions at any place in the stack.

JD

Subject: Re: main or procedure
Posted by Carsten Lechte on Thu, 21 Jun 2007 19:06:44 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> I tried this and it works well, prompting if you are stopped at a
> breakpoint, error, etc. Look for it in the next release, which is
> looking more and more like it will be 6.2 (some fixes since the Emacs
> 22 v6.1).

That sounds great!

chl
