Subject: Re: Locate an underflow
Posted by m.hadfield on Tue, 22 May 2001 22:05:41 GMT
View Forum Message <> Reply to Message

From: "Francesco" <francesco.spada@jrc.it>
> I've got this error
>
> % Program caused arithmetic error: Floating underflow
>
> Is it possible to know where it appens without putting a million of
> stop or breakpoint in my program?

Set the !EXCEPT system variable to 2.

From IDL 5.4 documentation:

!EXCEPT

An integer variable that controls when IDL checks for invalid mathematical computations (exceptions), such as division by zero. The three allowed values are:

Value Description

0 Never report exceptions.

1 Report exceptions when the interpreter is returning to an interactive prompt (the default).

2 Report exceptions at the end of each IDL statement. Note that this slows IDL by roughly 5% compared to setting !EXCEPT=1.

For more information on invalid mathematical computations and error reporting, see Math Errors. The value of !EXCEPT is used by the CHECK_MATH function to determine when to return errors. See CHECK_MATH for details.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--

Subject: Re: Locate an underflow
Posted by Ken Mankoff on Wed, 23 May 2001 02:58:02 GMT
View Forum Message <> Reply to Message

Hi Francesco,

I think you want to set the following system variable. I was playing with this today. Program execution will continue, but the standard error message (w/ line number included) should be printed out.

!except = 2

-k.

--
Ken Mankoff
LASP://303.492.3264
http://lasp.colorado.edu/~mankoff/

On Tue, 22 May 2001, Francesco wrote:

> Hi,
>
> I've got this error
>
> % Program caused arithmetic error: Floating underflow
>
> Is it possible to know where it appens without putting a million of
> stop or breakpoint in my program?
>
> Thank you
>
> Ciao
>
> Francesco
>

Subject: Re: Locate an underflow
Posted by thompson on Wed, 23 May 2001 15:08:46 GMT
View Forum Message <> Reply to Message

m.hadfield@niwa.cri.nz ("Mark Hadfield") writes:

> From: "Francesco" <francesco.spada@jrc.it>
>> I've got this error
>>

>> % Program caused arithmetic error: Floating underflow
>>
>> Is it possible to know where it appens without putting a million of
>> stop or breakpoint in my program?

> Set the !EXCEPT system variable to 2.

> From IDL 5.4 documentation:

>   !EXCEPT

>   An integer variable that controls when IDL checks for invalid
>   mathematical computations (exceptions), such as division by zero.
>   The three allowed values are:

>   Value Description

>   0 Never report exceptions.

>   1 Report exceptions when the interpreter is returning to an
>   interactive prompt (the default).

>   2 Report exceptions at the end of each IDL statement. Note that
>   this slows IDL by roughly 5% compared to setting !EXCEPT=1.

>   For more information on invalid mathematical computations and
>   error reporting, see Math Errors. The value of !EXCEPT is used by
>   the CHECK_MATH function to determine when to return errors. See
>   CHECK_MATH for details.

> ---
> Mark Hadfield
> m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
> National Institute for Water and Atmospheric Research

What I would really like is something that would differentiate between underflow errors and all other exceptions. I think that most users don't know what an underflow "error" is, and their initial reaction is that something bad is going on. I see that the IDL help now contains the following statement:

    A Note on Floating-Point Underflow Errors

    Floating-point underflow errors occur when a non-zero result is so close to zero that it cannot be expressed as a normalized floating-point number. In the vast majority of cases, floating-point underflow errors are harmless and can be ignored. For more information on floating-point numbers, see Accuracy & Floating-Point Operations.

In the old days, IDL didn't even bother to report underflow errors.

Even when you do know that underflow "errors" are (almost always) harmless, continuously getting these messages is highly annoying. I'd use !EXCEPT=0 to suppress underflow messages, except that it would also suppress other potentially more serious error messages as well.

How about something like the following for additional values for !EXCEPT:

Value Description

3 Suppress underflow errors, but report other exceptions when the interpreter is returning to an interactive prompt.

4 Suppress underflow errors, but report other exceptions at the end of each IDL statement.

William Thompson
