Subject: Re: Simple question regarding CATCH,/CANCEL
Posted by Michael Galloy on Tue, 28 Sep 2010 21:57:34 GMT
View Forum Message <> Reply to Message

On 9/28/10 3:28 PM, Paul van Delst wrote:
> Hello,
>
> I use CATCH error handling in just about all of my IDL procedures/functions.
>
> If no error occurs, do I need a
> CATCH, /CANCEL
> at the end of the pro/func? Or does exiting the pro/func cancel the error handling that was established on entry
> automatically?
>
> The documentation doesn't explicit state what happens, although the example does *not* have a "CATCH,/CANCEL" before the
> END (I also recall reading a post on this newsgroup stating that it was not necessary to CANCEL and error handler for
> normal termination).

In my experience, typical use of CATCH, /CANCEL is like:

```
catch, error
if (error ne 0L) then begin
  catch, /cancel
  ... handle error ..
endif
```

The CATCH, /CANCEL is used to turn off CATCH-ing errors in the "... handle error ..." part -- this would cause a recursive loop in the error handling code if an error was encountered in the error handling code itself.

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: Simple question regarding CATCH,/CANCEL
Posted by David Fanning on Tue, 28 Sep 2010 22:18:33 GMT
View Forum Message <> Reply to Message

mgalloy writes:

> In my experience, typical use of CATCH, /CANCEL is like:
>

> catch, error
> if (error ne 0L) then begin
> catch, /cancel
> ... handle error ..
> endif
>
> The CATCH, /CANCEL is used to turn off CATCH-ing errors in the "...
> handle error ..." part -- this would cause a recursive loop in the error
> handling code if an error was encountered in the error handling code itself.

Outside of debugging code, this is about the only way I
ever use it, too.

I could see using it inside a function that had an
ON_ERROR, 2 condition that I wanted to temporarily
override to catch an anticipated error in the function.
When I got past the danger spot I might cancel the
Catch so that the normal RETURN from the function was
in effect.

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: Simple question regarding CATCH,/CANCEL
Posted by Paul Van Delst[1] on Tue, 28 Sep 2010 22:51:24 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> mgalloy writes:
>
>> In my experience, typical use of CATCH, /CANCEL is like:
>>
>> catch, error
>> if (error ne 0L) then begin
>> catch, /cancel
>> ... handle error ..
>> endif
>>
>> The CATCH, /CANCEL is used to turn off CATCH-ing errors in the "...

>> handle error ..." part -- this would cause a recursive loop in the error
>> handling code if an error was encountered in the error handling code itself.
>
> Outside of debugging code, this is about the only way I
> ever use it, too.

O.k., good to know.

I can now delete all the superfluous "CATCH,/CANCEL"'s in my code.

> I could see using it inside a function that had an
> ON_ERROR, 2 condition that I wanted to temporarily
> override to catch an anticipated error in the function.
> When I got past the danger spot I might cancel the
> Catch so that the normal RETURN from the function was
> in effect.

Hmm. I no longer use ON_ERROR because CATCH seems to, well, catch everything error-wise.

I do the following for temporary overrides via a ubiquitous Debug keyword:

```
 IF ( KEYWORD_SET(Debug) ) THEN BEGIN
   MESSAGE, '--> Entered.', /INFORMATIONAL
   MsgSwitch = 0
 ENDIF ELSE BEGIN
   CATCH, Error_Status
   IF ( Error_Status NE 0 ) THEN BEGIN
     CATCH, /CANCEL
     MESSAGE, !ERROR_STATE.MSG
   ENDIF
   MsgSwitch = 1
 ENDELSE
```

(with similar for functions)

All my subsequent error message output does something like

```
  MESSAGE, 'an error has occurred', NONAME=MsgSwitch, NOPRINT=MsgSwitch
```

If I set Debug, the error msg is output and execution stops so I can interactively inquire what happened.
If I don't set Debug, the error msg is output and the error handler takes over and the error bubbles up the call chain.

I think I'm doing something twice here that doesn't need doing.... or I'm leaving something out... can't remember which.
But it works pretty well. :o)

cheers,

paulv