Subject: CONTINUE from function within loop
Posted by spluque on Wed, 13 Nov 2013 05:19:43 GMT
View Forum Message <> Reply to Message

Hi,

I read the CATCH and ON_ERROR help pages, but I can't see how to handle the following situation. Say we have a FOR loop where a function is called for each iteration:

```
FOR i=0, 10 DO BEGIN
   print, myfun(i)
ENDFOR
```

I would like MYFUN to continue with the next iteration if it runs into an error. Naively, this should show what I'm looking for:

```
FUNCTION MYFUN, X

   [BODY]

    IF [condition] THEN BEGIN
       message, 'A problem with X', /informational
       CONTINUE
    ENDIF

    [REST IF OK]

    RETURN, [something]

END
```

But, of course, CONTINUE only works within an FOR/ENDFOR block, not this way. How does one accomplish this?

Thanks,
Seb

Subject: Re: CONTINUE from function within loop
Posted by Craig Markwardt on Wed, 13 Nov 2013 05:40:34 GMT
View Forum Message <> Reply to Message

On Wednesday, November 13, 2013 12:19:43 AM UTC-5, Sebastian Luque wrote:
> Hi,
>
>
>
> I read the CATCH and ON_ERROR help pages, but I can't see how to handle the following

situation. Say we have a FOR loop where a function is called for each iteration:

```
>
>
>
>  FOR i=0, 10 DO BEGIN
>
>     print, myfun(i)
>
>  ENDFOR
>
>
>
```

> I would like MYFUN to continue with the next iteration if it runs into an error. Naively, this should show what I'm looking for:

```
>
>
>
>  FUNCTION MYFUN, X
>     [BODY]
>      IF [condition] THEN BEGIN
>          message, 'A problem with X', /informational
>          CONTINUE
>      ENDIF
>      [REST IF OK]
>      RETURN, [something]
>  END
>
```

> But, of course, CONTINUE only works within an FOR/ENDFOR block, not this way. How does one accomplish this?

You're right, because CONTINUE only works within the same lexical scope.

You could do this by putting a CATCH within your main loop, which will catch your MESSAGE within MYFUN. That's a little dangerous because it will catch *every* error though.

A better practice is to return a separate status variable MYFUN indicating success or failure, and have the main loop check the variable and decide what to do.

Something like this...

```
function myfun, x, status=status
  status = -1
  ;; Failure case
  if [condition] then return, !values,d_nan
  [ rest of MYFUN ]
  ;; Success case
  status = 1
  return, [something]
end
```

```
for i = 0, 10 do begin
  y = myfun(x, status=status)
  if status LT 0 then continue
  print, y
endfor
```

Craig

Subject: Re: CONTINUE from function within loop
Posted by Moritz Fischer on Wed, 13 Nov 2013 06:32:33 GMT
View Forum Message <> Reply to Message

Hi,
If this was possible, it would make your myfun routine very hard to reuse.
Why don't you add a keyword to MYFUN ?

Replace the loop by

```
%<---------------------
FOR i=0, 10 DO BEGIN
   tmp = myfun(i, FAILED=FAILED, MSG=MSG)
   IF FAILED THEN BEGIN
     message, MSG, /INFO
     CONTINUE
   ENDIF ELSE print, tmp
ENDFOR
%<---------------------
```

where

```
%<---------------------
FUNCTION MYFUN, X, FAILED = FAILED, MSG=MSG

  [BODY]

  FAILED = [condition]

  IF FAILED THEN BEGIN
    MSG = 'A problem with X',
    RETURN, !NULL
  ENDIF

  [REST IF OK]

  RETURN, [something]
  END
%<---------------------
```

But this does not work if MYFUN actually runs into an error (but only
if [condition] is true) .
What's wrong with putting thte following into the FOR loop?

```
%<---------------------
CATCH, err
IF err NE 0 THEN BEGIN
   message, !error_state.message, /INFO
   CATCH, /cancel
   CONTINUE
ELSE print, myfun(i)
%<---------------------
```

I think (hope) this is the only way you can jump to another level.

cheers

Am 13.11.2013 06:19, schrieb Sebastian Luque:
> Hi,
>
> I read the CATCH and ON_ERROR help pages, but I can't see how to
> handle the following situation. Say we have a FOR loop where a
> function is called for each iteration:
>
> FOR i=0, 10 DO BEGIN print, myfun(i) ENDFOR
>
> I would like MYFUN to continue with the next iteration if it runs
> into an error. Naively, this should show what I'm looking for:
>
> FUNCTION MYFUN, X
>
> [BODY]
>
> IF [condition] THEN BEGIN message, 'A problem with X',
> /informational CONTINUE ENDIF
>
> [REST IF OK]
>
> RETURN, [something]
>
> END
>
> But, of course, CONTINUE only works within an FOR/ENDFOR block, not
> this way. How does one accomplish this?
>

> Thanks, Seb
>

Subject: Re: CONTINUE from function within loop
Posted by spluque on Wed, 13 Nov 2013 15:21:37 GMT
View Forum Message <> Reply to Message

On Tuesday, November 12, 2013 11:40:34 PM UTC-6, Craig Markwardt wrote:
> On Wednesday, November 13, 2013 12:19:43 AM UTC-5, Sebastian Luque wrote:
>
>> Hi,
>
>>
>
>>
>
>>
>
>> I read the CATCH and ON_ERROR help pages, but I can't see how to handle the following situation. Say we have a FOR loop where a function is called for each iteration:
>
>>
>
>>
>
>>
>
>> FOR i=0, 10 DO BEGIN
>
>>
>
>>   print, myfun(i)
>
>>
>
>> ENDFOR
>
>>
>
>>
>
>>
>
>> I would like MYFUN to continue with the next iteration if it runs into an error. Naively, this should show what I'm looking for:
>
>>

```
>
>>
>
>>
>
>>  FUNCTION MYFUN, X
>
>>     [BODY]
>
>>      IF [condition] THEN BEGIN
>
>>          message, 'A problem with X', /informational
>
>>          CONTINUE
>
>>      ENDIF
>
>>      [REST IF OK]
>
>>      RETURN, [something]
>
>>  END
>
>>
>
```
>> But, of course, CONTINUE only works within an FOR/ENDFOR block, not this way. How does one accomplish this?

>

>

>

> You're right, because CONTINUE only works within the same lexical scope.

>

>

>

> You could do this by putting a CATCH within your main loop, which will catch your MESSAGE within MYFUN. That's a little dangerous because it will catch *every* error though.

>

>

>

> A better practice is to return a separate status variable MYFUN indicating success or failure, and have the main loop check the variable and decide what to do.

>

>

>

> Something like this...

```
>
>  function myfun, x, status=status
>
```

```
>   status = -1
>
>   ;; Failure case
>
>   if [condition] then return, !values,d_nan
>
>   [ rest of MYFUN ]
>
>   ;; Success case
>
>   status = 1
>
>   return, [something]
>
> end
>
>
>
> for i = 0, 10 do begin
>
>   y = myfun(x, status=status)
>
>   if status LT 0 then continue
>
>   print, y
>
> endfor
>
```

This is a very clean solution!

Thanks,
Seb