Subject: Re: Stop and return to caller
Posted by Paul Van Delst[1] on Thu, 22 Jul 2010 14:00:35 GMT
View Forum Message <> Reply to Message

I'm a little bit confused by your terminology. On error, do you want a procedure to stop execution (i.e. like an IDL
STOP), or do you want to return to the caller (i.e. an IDL RETURN)? You can't do both (well, you can, but it requires
user interaction).

Assuming you're simply talking about error handling, this is the sort of thing I do:

```
pro my_pro, a, b, debug=debug

  ; Setup error handler
  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
      ; ...do any other procedure cleanup...
      message, !error_state.msg, /informational
      return
    endif
    msgswitch = 1
  endelse

  ; Do procedure calcs
  c = a + b
  if (c lt 0) then $
    message, 'c < 0!', noname=msgswitch, noprint=msgswitch

end
```

Running the above with input to create the error and without the DEBUG switch set:

```
IDL> my_pro, -2, -3
% MY_PRO: c < 0!
IDL> help
% At $MAIN$
Compiled Procedures:
   $MAIN$  MY_PRO

Compiled Functions:
```

I.e. the error is handled and control returns to the caller.

Running WITH the debug switch set gives the following:

```
IDL> my_pro, -2, -3,/debug
% MY_PRO: --> Entered.
% MY_PRO: c < 0!
% Execution halted at: MY_PRO            20 /scratch/my_pro.pro
%                      $MAIN$
IDL> help
% At MY_PRO            20 /scratch/my_pro.pro
%    $MAIN$
A               LONG      =          -2
B               LONG      =          -3
C               LONG      =          -5
DEBUG           INT       =       1
ERROR_STATUS    UNDEFINED = <Undefined>
MSGSWITCH       INT       =       0
Compiled Procedures:
   $MAIN$  COLORS      MY_PRO

Compiled Functions:
```

That is, the error is still handled, but execution stops where the error occurred allowing you to inspect variables to
figure out what happened.

The error handling code is a lot of repetitive typing, particularly if the procedure/function is only a couple of lines.
I tend to put these error handlers in include files so they're easy to (re)use, e.g.

```
pro my_pro, a, b, debug=debug

  ; Setup error handler
  @procedure_error_handler

  ; Do procedure calcs
  c = a + b
  if (c lt 0) then $
    message, 'c < 0!', noname=msgswitch, noprint=msgswitch

end
```

where all the error handling code of the previous example is now in a file called "procedure_error_handler.pro". There
are some nit-picky, not-so-pretty details in doing this in a generic way, but I won't go into those

here other than
chant my mantra: convention over configuration.

cheers,

paulv

Deckard++; wrote:
> Hi,
>
> I have a small question for which I haven't found a pleasant solution.
> I am looking for a way to stop the execution of a procedure and return
> to the level of the programme that called that procedure. Currently I
> have something like this:
>
> pro my_pro,a,b
> on_error,2
>
> c = a + b
> if (c lt 0) then message,'c < 0! Stopping...'
> end
>
> It works, but the main problem is that since it is generated as an
> error, it outputs all the calling stack with line numbers and so on. I
> would like to do something cleaner, like for instance the "stop"
> procedure which outputs only the current position in the execution.
>
> Thanks a lot in advance,
>
> -- Arthur;
>