Subject: Allowing interruption and background processing in MS-Windows
Posted by HADFIELD[1] on Mon, 04 Apr 1994 00:28:06 GMT
View Forum Message <> Reply to Message

A recent message pointed out that a procedure running in IDL for MS-Windows can't be interrupted by Ctrl-C. A related problem under MS-Windows is that IDL generally does not allow other Windows applications to execute while an IDL procedure is running. There are two reasons for this:

1. Windows supports cooperative multi-tasking and requires each application to explicitly yield control in order for other applications to execute. IDL does not do this.

2. When running a procedure IDL typically displays an hourglass cursor and traps all keyboard & mouse input.

I don't generally find this a BIG problem because most operations in IDL are so fast. The conspicuous exception is an iterative task like reading data line-by-line from a large ASCII file. I have found a method of persuading IDL to share the computer during this sort of task. The procedure YIELD (below) can be called once every few iterations. It does two things:

1. It calls the Windows API function Yield to yield control to other applications.

2. It displays a button widget offering the user the opportunity to abort or suspend the task.

Incidentally these two aspects to YIELD's operation are complementary. Just calling the API function without displaying the widget means that other Windows appilcations (eg the Clock) will operate but the user can't get at them. Just displaying the widget without calling the API function allows the user to switch to other applications but then the button widget can't be activated!

As an example of the use of YIELD, the function NLINES (see comments in YIELD.PRO below), originally written by F Knight, counts the number of lines in a file. On my system it takes 1.5 s per 1000 lines. If YIELD is called at every 30th line the time to process the file is increased by 10% (if the machine is idle) but other applications will run satisfactorily.

YIELD handles the widget initialisation as necessary. At the end of the task YIELD should be called with the keyword DESTROY to destroy the widget . Alternatively the widget can be closed via its window control button. It doesn't matter all that much if it's left open--it's just that any button-click events will be detected the next time YIELD is called.

-- YIELD.PRO -- -----

```
;+
; NAME:
;   YIELD
;
; PURPOSE:
;   Yields control to the Windows operating system to allow background processing
;   & allows the user to abort or suspend execution.
;
; CATEGORY:
;   Utilities.
;   MS Windows.
;
; CALLING SEQUENCE:
;   YIELD
;
; INPUTS:
;   None
;
; INPUT KEYWORDS
;   DESTROY     If set, Yield destroys the widget base.
;
; OUTPUTS:
;   None.
;
; SIDE EFFECTS:
;   A button menu widget is displayed. Other Windows processes are given an opportunity
;   to execute.
;
; DEPENDENCIES:
;   Has an effect only on MS-Windows. May require IDL version >= 3.5 for
;   CALL_EXTERNAL to work properly.
;
; PROCEDURE:
;   Yield calls widget routines as necessary to maintain a button widget offering the
;   choices 'Abort', 'Suspend' and 'Resume'. (The last is initially insensitive.)
;   At each call, Yield checks for widget events and also calls the Windows API
;   function "Yield" to allow other processes to execute. If no widget events are
;   detected, Yield returns to the calling program. If 'Abort' is selected
;   Yield writes a message & terminates. If 'Suspend' is selected, Yield sensitises the
;   'Resume' button and goes into a loop waiting for it to be pressed.
;
; EXAMPLE:
;
;   Calls to yield should be distributed throughout lengthy computations.
;   Yield takes approximately 8 ms to execute on an otherwise idle 486/66.
;   (Most of that time is taken for the "Yield" API function.) So to give reasonable
;   response in other applications without slowing IDL too much it should be
;   called approx once every 0.2 sec. There should be a final call with the
;   DESTROY keyword to remove the widget. (Otherwise it should be manually closed.)
;
;   The following function (based on one by F Knight) counts the lines
;   in a file and optionally calls Yield during processing:
;
;   function nlines, file, yield=yield
;
;       nl = 0L  &  tmp=''
;       openr,lun,file,/get_lun
;       on_ioerror,NOASCII
;       while not eof(lun) do begin
;           readf,lun,tmp
;           nl = nl + 1
;           if keyword_set(yield) then if (nl mod 30) eq 0 then yield
;       endwhile
;       close,lun
;       free_lun,lun
;       NOASCII:
;       if keyword_set(yield) then yield, /destroy
;       return,nl
;
;   end
;
;   Reading a 3141 line file takes 5.3 second with the YIELD keyword set
;   and 4.6 without. (If the references to the YIELD keyword are
;   removed entirely, it still takes 4.6 s.)
;
; MODIFICATION HISTORY:
;   Mark Hadfield, 3 April 1994:
;       Created.
;-

pro yield, destroy=destroy

    common yield_widget, base_id, button_id

    on_error,2

    if strlowcase(!version.OS) ne 'windows' then return

    if n_elements(base_id) eq 0 then base_id = 0L

    if not keyword_set(destroy) then begin

        if not widget_info(base_id,/valid) then $
            xmenu, ['Abort','Suspend','Resume'], base=base_id, button=button_id, /row, title='Yield'
        if not widget_info(base_id,/real) then begin
           widget_control, base_id, /realize
           widget_control, button_id(2), sensitive=0
        endif

        event = widget_event(base_id,/nowait)

        OK = call_external("kernel.exe","Yield")

        if event.id ne 0 then begin

           case event.id of

           button_id(0):   begin
                          widget_control, base_id, /destroy
                          message, "Operation aborted. " + $
                                 "You may have to press RETALL to return to the main program level."
                       end

           button_id(1):   begin
                          message, /inform, "Operation suspended."
                          widget_control, button_id(0), sensitive=0
                          widget_control, button_id(1), sensitive=0
                          widget_control, button_id(2), sensitive=1
                          event = widget_event(base_id,/nowait)
                          while event.id ne button_id(2) do begin
                             event = widget_event(base_id,/nowait)
                             OK = call_external("kernel.exe","Yield")
                          endwhile
                          widget_control, button_id(0), sensitive=1
                          widget_control, button_id(1), sensitive=1
                          widget_control, button_id(2), sensitive=0
                          message, /inform, "Operation resumed."
                       end

           else: message, "Unexpected event from Yield widget."

           endcase

        endif

     endif else begin

        if widget_info(base_id,/real)  then widget_control, base_id, /destroy

     endelse

  end
```

```
============================================================= ==========
```

Mark Hadfield hadfield@greta.niwa.cri.nz
NIWA Marine (Taihoro Nukurangi) NIWA.GRETA:HADFIELD
Wellington, New Zealand