Subject: Re: IDLVM and retall
Posted by JD Smith on Thu, 07 Sep 2006 08:57:12 GMT
View Forum Message <> Reply to Message

On Wed, 06 Sep 2006 23:21:54 -0700, JD Smith wrote:

> On Wed, 06 Sep 2006 21:30:08 +0200, Fï¿½LDY Lajos wrote:
>
>>
>> On Wed, 6 Sep 2006, JD Smith wrote:
>>
>>> On Tue, 05 Sep 2006 17:31:04 +0200, Fï¿½LDY Lajos wrote:
>>>
>>>>
>>>> On Mon, 4 Sep 2006, JD Smith wrote:
>>>>
>>>> > [quoted text muted]
>>>>
>>>> probably a CATCH in main helps. It works in IDL 6.2 (linux).
>>>
>>> But that blocks my active command line.
>>>
>>> Thanks,
>>>
>>> JD
>>>
>>
>> your original problem was RETALL not working in the IDL VM, where you have
>> no command line. You can use RETALL in real IDL, CATCH in the VM. There is
>> a /VM test in LMGR. I know, the code will be ugly, but will work.
>>
>
> I thought of that after I responded, but (correct me if wrong), I don't
> think you can even have a main level routine in the VM, since your entry
> routine must be a named procedure.

I eventually resorted to something along these lines:

```
pro test_vm_event, ev
  a=dialog_message('Testing RETALL',/ERROR)
  if lmgr(/VM) || lmgr(/RUNTIME) then message,'TEST-VM-ERROR',/NOPRINT else $
     retall
end

pro test_vm
  command_line=~lmgr(/VM) && ~lmgr(/RUNTIME)
  b=widget_base(/ROW)
  but=widget_button(b,VALUE='Do It!')
```

```
  widget_control, b,/realize
  if ~command_line then begin
    XManager,CATCH=0
    catch,err
    ;; Just keep processing events
    if err ne 0b then begin
      XManager
      return
    endif
  endif
  XManager,'test_vm',b,NO_BLOCK=command_line
end
```

I.e. using a blocking XManager call if there is no command line. As long as the first entry call to XManager in the VM program is blocking, all subsequent calls (starting other related widgets, for exmaple) will use the XManager event loop, by-passing the (quasi-broken, in my opinion) "active command line" widget processing equivalent in the VM.

Unhandled errors will be sent all the way out to this top level routine, which can ignore the error and kick start event processing again with an argument-less call to XMANAGER. If this (or any subsequent) nested XMANAGER returns, return from the entire program to avoid double calls to the original entry XMANAGER.

The only other ingredient is turning off XManager's default catch mechanism, and using MESSAGE to throw an error instead of RETALL. It's not elegant, but it does the job.

JD

P.S. BTW, Fï¿½ldy, in your original suggestion, the main level CATCH part was never being compiled: when the compiler gets to TEST_VM, it stops. The ON_ERROR,1 was what was doing the trick for you (only in commmand-line enabled runtimes).
