Subject: really dumb widget question
Posted by Mark Fardal on Fri, 14 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,

When IDL is stopped within a procedure, a non-blocking widget will
still listen to events. But it won't act on them. Why not? Is there
any way to make it act on them?

E.g., if you run from the main level a procedure with a stop
statement, call XLOADCT, and hit one of the color table choices, the
color table will change to what you requested, but not until you
return to the main level. Calling LOADCT of course gets you a changed
color table right away.

I guess my problem is understanding how the widget event loop interacts
with the command line. Widgetheads, please help.

thanks,
Mark Fardal
UMass

Subject: Re: really dumb widget question
Posted by Paddy Leahy on Tue, 03 Sep 2013 14:37:32 GMT
View Forum Message <> Reply to Message

On Friday, April 14, 2000 8:00:00 AM UTC+1, Mark Fardal wrote:
> (David Fanning) writes:
>
>>> When IDL is stopped within a procedure, a non-blocking widget will
>>> still listen to events. But it won't act on them. Why not? Is there
>>> any way to make it act on them?
>>
>> I presume because STOP means STOP. If it meant MOSTLY STOP,
>> I expect RSI would hear about it. :-)
>>
>> I would try .CONTINUE to get them going again.
>
> Hi,
>
> Well, I did know that much. But as I noted, you have to wait until
> returning to the main level. (Or is it just leaving the procedure with
> the STOP? Didn't test.) That isn't always what I want.
>
> Say you write two versions of a procedure. One takes user input from
> the command line, the other takes it from a non-blocking widget. If

> you stop at a certain point and need to use the procedure before going
> on, you can use the command-line version. But the GUI version is
> useless, unless you rewrite it as a blocking widget.
>
> Non-blocking widgets introduce parallel threads of execution to an
> environment that doesn't otherwise use threads. I would naively
> expect a STOP to stop the thread it's in, but apparently it stops all
> of them--except for the one that _records_ widget events for later
> processing. I'm basically wondering if one thread can tell another
> thread to go ahead and execute.
>
> Mark

Here we are in 2013, IDL 8.2 and this is still happening. I think it is a bug, not a feature. The symptom is this: if a script has been run and hit a stop statement, then control returns to the command line. If you don't bother to .continue or retall or equivalent, you can happily run any normal procedure and get the expected results. But if you have a non-blocking widget program running under XMANAGER, the event loop is halted. It re-starts again (with the accumulated requested events being acted on) as soon as you .continue etc. If you start off a widget utility after some script has hit a stop, the widget procedure works (creates the widget etc) up until the point that xmanager is invoked, then halts, leaving you with a frozen widget until you .continue the script. Typing "STOP" on the command line does not have this effect, it has to be in a procedure: any procedure (or function), typically with no connection at all with the widget you are trying to run in the background.

How is that not a bug?

Subject: Re: really dumb widget question
Posted by bill.dman on Tue, 03 Sep 2013 17:13:03 GMT
View Forum Message <> Reply to Message

On Tuesday, September 3, 2013 10:37:32 AM UTC-4, Paddy Leahy wrote:
> On Friday, April 14, 2000 8:00:00 AM UTC+1, Mark Fardal wrote:
>
>> (David Fanning) writes:
>
>>
>
>>>> When IDL is stopped within a procedure, a non-blocking widget will
>
>>>> still listen to events. But it won't act on them. Why not? Is there
>
>>>> any way to make it act on them?
>
>>>
>
>>> I presume because STOP means STOP. If it meant MOSTLY STOP,

>
>>> I expect RSI would hear about it. :-)
>
>>>
>
>>> I would try .CONTINUE to get them going again.
>
>>
>
>> Hi,
>
>>
>
>> Well, I did know that much. But as I noted, you have to wait until
>
>> returning to the main level. (Or is it just leaving the procedure with
>
>> the STOP? Didn't test.) That isn't always what I want.
>
>>
>
>> Say you write two versions of a procedure. One takes user input from
>
>> the command line, the other takes it from a non-blocking widget. If
>
>> you stop at a certain point and need to use the procedure before going
>
>> on, you can use the command-line version. But the GUI version is
>
>> useless, unless you rewrite it as a blocking widget.
>
>>
>
>> Non-blocking widgets introduce parallel threads of execution to an
>
>> environment that doesn't otherwise use threads. I would naively
>
>> expect a STOP to stop the thread it's in, but apparently it stops all
>
>> of them--except for the one that _records_ widget events for later
>
>> processing. I'm basically wondering if one thread can tell another
>
>> thread to go ahead and execute.
>
>>
>
>> Mark

>
>
>
> Here we are in 2013, IDL 8.2 and this is still happening. I think it is a bug, not a feature. The symptom is this: if a script has been run and hit a stop statement, then control returns to the command line. If you don't bother to .continue or retall or equivalent, you can happily run any normal procedure and get the expected results. But if you have a non-blocking widget program running under XMANAGER, the event loop is halted. It re-starts again (with the accumulated requested events being acted on) as soon as you .continue etc. If you start off a widget utility after some script has hit a stop, the widget procedure works (creates the widget etc) up until the point that xmanager is invoked, then halts, leaving you with a frozen widget until you .continue the script. Typing "STOP" on the command line does not have this effect, it has to be in a procedure: any procedure (or function), typically with no connection at all with the widget you are trying to run in the background.
>
>
>
> How is that not a bug?

Here's my workaround in a one line module I named runwids.pro:

print,'@runwids: Running widget events. Press control-C to stop.' & while 1 do begin & wait, 0.05 & _$__ = widget_event(/NOWAIT) & endwhile

After STOP, @runwids reactivates widgets until control-C is pressed,
which returns control to the command line in the context of your STOP. Note,
you should wait until widget event code or iTool calculations are complete
before doing control-C, or you can end up in the context of those codes. If this happens, just type .continue, and control-C again.

It's awkward but occasionally worth while.

Subject: Re: really dumb widget question
Posted by chris_torrence@NOSPAM on Wed, 04 Sep 2013 20:08:45 GMT
View Forum Message <> Reply to Message

Hi Paddy,

Well, I can tell you that this is not a bug, it is by design. I just looked in the C code, and it says if the interpreter is "idle", but you're at a stop or a breakpoint, then *only* process widget events if you are at $main.

I imagine that this must have been done so that if you were debugging the event handler of a widget program, that you didn't end up calling the event handler again and hitting the same stop.

Now, I just commented out that line of C code, and it works "as expected". I can now stop within a procedure and do xloadct or p=plot() and I can interact with the widgets. This actually fixes another bug where you try to use new graphics (or itools) to look at your data while debugging.

But... what if you really are trying to debug your widget app, and it keeps sending widget events to your event handler? Maybe that is okay - you just shouldn't click on the button that sends the event?

Thoughts?

-Chris
ExelisVIS

Subject: Re: really dumb widget question
Posted by David Fanning on Wed, 04 Sep 2013 20:22:45 GMT
View Forum Message <> Reply to Message

Chris Torrence writes:

> But... what if you really are trying to debug your widget app, and it keeps sending widget events to your event handler? Maybe that is okay - you just shouldn't click on the button that sends the event?

Or move your cursor at all. ;-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: really dumb widget question
Posted by Paddy Leahy on Sat, 07 Sep 2013 17:59:45 GMT
View Forum Message <> Reply to Message

On Wednesday, 4 September 2013 21:08:45 UTC+1, Chris Torrence wrote:
> Hi Paddy,
>
>
>
> Well, I can tell you that this is not a bug, it is by design. I just looked in the C code, and it says if

the interpreter is "idle", but you're at a stop or a breakpoint, then *only* process widget events if you are at $main.
>
>
>
> I imagine that this must have been done so that if you were debugging the event handler of a widget program, that you didn't end up calling the event handler again and hitting the same stop.
>
>
>
> Now, I just commented out that line of C code, and it works "as expected". I can now stop within a procedure and do xloadct or p=plot() and I can interact with the widgets. This actually fixes another bug where you try to use new graphics (or itools) to look at your data while debugging.
>
>
>
> But... what if you really are trying to debug your widget app, and it keeps sending widget events to your event handler? Maybe that is okay - you just shouldn't click on the button that sends the event?
>
>
>
> Thoughts?
>
>
>
> -Chris
>
> ExelisVIS

Hi Chris,
Thanks for looking into this. I guess I see the problem. Two thoughts:

Document this behaviour, both in the description of XMANAGER and in the general description of widget programming. I suppose it is alluded to in the page on Debugging Widget Applications, which advises a RETALL to restart stopped widgets, but it could be made clear that this applies even if the stopped program has nothing to do with the widget.

The other thought is that it must be possible to handle this more elegantly. Ideally, event processing would be turned off only if the stopped/crashed routine was ultimately called by the widget. If this is not possible, a switch in XMANAGER to turn this behaviour on or off might work. As David Fanning points out, some widgets can generate a lot of events very quickly which I guess could be a problem if event processing was enabled at the time.

regards,
Paddy

Subject: Re: really dumb widget question
Posted by chris_torrence@NOSPAM on Mon, 09 Sep 2013 22:09:23 GMT
View Forum Message <> Reply to Message

Okay, this has been fixed for IDL 8.3. Here's the documentation for the "What's New":

Event handling while debugging
In the past, IDL would not sent widget events when you were stopped within a routine. Now, by default, IDL sends widget events even when stopped within a routine. This allows you to use graphics and widget applications while debugging.
There is a new system variable, !DEBUG_PROCESS_EVENTS, that can be set to 0 to disable this behavior, or to 1 to enable this behavior. The default value is 1.

Cheers,
Chris
ExelisVIS