Subject: Re: Returning result from a widget program.
Posted by davidf on Fri, 10 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Hi Folks,

I've just returned from a short holiday where I have resolved, once again, to give up this strange IDL fetish I have and get on with the rest of my life. But, alas, my work remains unfinished...

Imanol Echave (ccaeccai@sc.ehu.es) writes:

> I'm writting an IDL function that works with widgets. The user calls an IDL
> function that shows an widget interface to input data. When the user pushes the
> "OK" button the function has to return a result that depends on the input data.
> My problem is where to store this result to return it. I can't use the UVALUE of
> the widgets because when I want to return the result the widgets are destroyed.
> Any advice?

And he receives advice like this. Sigh...

> If you need the parameter var in an another program (or in an
> another subrutine) I suggest to put it in a common block.

Even Mirko Vukovic (who I sent my book to, for goodness sake!) recommends a common block.

Only David Foster offers a word of caution about common blocks in widget programs.

Let me put it this way. From time to time widget programs are very useful. So useful that you might even want to use several instances of that program at the same time. (A dialog widget function that collects information about which file to open comes to mind, or a program to load colors in a particular window, or a program that processes images in a particular way, etc, etc. In fact, just about every widget program I write meets the criteria.)

But if you use a common block in that widget program, you can only run ONE INSTANCE of that program at any particular time. If you run more than one instance, your programs will not work properly. Period. This makes common blocks in widget programs a lousy choice in my humble opinion.

So it is important to know how to write widget programs WITHOUT common blocks. In this case, Mr. Echave is absolutely correct. He cannot use the UVALUE to store information he

collects from the user. Because by the time he collects it
and destroys the modal widget, the user value (and everything
stored there) is gone. He can't get it back to return it
as the result of the function. He must store the information
in a global location that is *external* to the widgets used
to create the program. A pointer location is *exactly* what
is called for, as David Foster indicates.

The last several lines of the data collection function
might look like this:

```
  dataPtr = Ptr_New({cancel:1})
```

(My data pointers usually point to a structure that contains
the information I hope to collect from the user. This might
be the name of a data file, the type of data stored there,
the size of the data, etc. I like to have a field in that
structure that tells me if the user hit the CANCEL button
on my dialog. If so, the CANCEL field in the structure is set
to 1. When I set up the data pointer I usually turn this
CANCEL flag ON so that all I really have to worry about is
if the user hit the OK or ACCEPT button. This keeps me from
coming to grief if the user just kills the widget with his
or her mouse instead of using the thoughtfully provided
buttons.)

```
  info = {dataPtr:dataPtr, ..., ...}
  Widget_Control, tlb, Set_UValue=info, /No_Copy
  XManager, 'example', tlb   ; Modal widget blocks here.

     ; User killed widget. Get data and return it.

  data = *dataPtr
  Ptr_Free, dataPtr
  IF data.cancel THEN RETURN, -1 ELSE RETURN, data
  END
```

You can see how this works in more detail by looking at
the programs GETIMAGE or GETDATA from my anonymous ftp
site:

```
   ftp://ftp.dfanning.com/pub/dfanning/outgoing/coyote/getimage .pro
   ftp://ftp.dfanning.com/pub/dfanning/outgoing/coyote/getdata. pro
```

If you have trouble with the programs (they are well documented),
you can read the last two chapters in my book. They talk about
how to build both modal and non-modal widget dialogs without using
common blocks.

If I don't hear the words "common block" mentioned here for
at least two weeks, I know it will be safe to retire. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/
