Subject: Returning result from a widget program.
Posted by Imanol Echave on Thu, 09 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Hi people:

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?

Subject: Re: Returning result from a widget program.
Posted by Eddie Breeveld on Mon, 13 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

But common blocks are clear and easy! Your pointer example
(this is me showing some prejudice!) is unclear and more importantly
easy to get wrong.

Eddie Breeveld

David Foster wrote:
>
> mirko_vukovic@notes.mrc.sony.com wrote:
>>
>>   Imanol Echave <ccaeccai@sc.ehu.es> wrote:
>>> Hi people:
>>>
>>>   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. Any advice?
>>
>> common block, pointer variable?
>>
>> You can setup a widget cleanup routine (and frankly, I forget their syntax
>> now), where you can do that kind of stuff.
>>
>> mirko
>>
>
> This is definitely the easiest approach, but it's messy and you'll

> regret it later. There is almost always a way to avoid common blocks,
> like my recent reply to Imanol.
>
> Dave
>

--
Edward Breeveld MSSL/UCL, Holmbury St. Mary, Dorking, Surrey RH5 6NT, UK
e.breeveld@physics.org tel: +44 (0)1483 204178/267632 fax: +44 (0)1483 278312

Subject: Re: Returning result from a widget program.
Posted by Vap User on Mon, 13 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

mirko_vukovic@notes.mrc.sony.com writes:

It's unclear exactly what you're asking. If the widget program you call can be cast as a function, you can return the result of whatever calculations/changes to data/... occur as the function result. Take a look at Dave Fanning's PS_FORM.pro (http://www.dfanning.com and follow your nose) for an example of this sort of thing.

If you're talking about a popup widget that can't, for some reason, return the result in the return value, pass it a pointer to your data and whatever space you might want to reserve and have the widget store the pointer in a structure in its UVALUE. The pointer to the modified data will be available to the calling program after the popup has been destroyed.

If at all possible, go with option 1.

William Daffer

>
> In article <35A4BF4B.5FE452DE@sc.ehu.es>,
> Imanol Echave <ccaeccai@sc.ehu.es> wrote:
>> Hi people:
>>
>> 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?
>>
>
> (I guess it is my turn to show what I've learned about widget programming)
>
> common block, pointer variable?
>
> You can setup a widget cleanup routine (and frankly, I forget their syntax
> now), where you can do that kind of stuff.
>

  Xmanager, 'name', ID , cleanup='name_of_cleanup_routine'

This overrides whatever routine was specified using the 'Kill_Notify'
keyword to Widget_Control.

> mirko
>
> you
>
> -----== Posted via Deja News, The Leader in Internet Discussion ==-----
> http://www.dejanews.com/rg_mkgrp.xp   Create Your Own Free Member Forum

--
I don't speak for JPL, it doesn't speak for me.
Well, not all the time, at least.
William Daffer <vapuser@haifung.jpl.nasa.gov>

Subject: Re: Returning result from a widget program.
Posted by Martin Schultz on Tue, 14 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

R. Bauer wrote:
> [...]

> I agree to you in not using common blocks in widgets in most cases.But how can I
> handle 60MB of data without a common block. If I use the set_uval and get_uval this
> 60MB will be copied for a while and it takes a lot of time and memory. I don't found a
> key to make a temporary copy using widgets. May be I am missing something.
>
> Any ideas?

>
> Reimar
>
Hi Reimar,

you know about the /NO_COPY keyword to widget_control, don't you?
The same keyword exists for PTR_NEW. True, you need to be a little more
careful when you use them, but they are meant to resolve exactly the
problem you mention.

Viele Gruesse,
Martin.

--
-- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
109 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
Internet-homepage: http://www-as.harvard.edu/people/staff/mgs/
-- -------

Subject: Re: Returning result from a widget program.
Posted by R. Bauer on Tue, 14 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> 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
>
>

I agree to you in not using common blocks in widgets in most cases.But how can I handle 60MB of data without a common block. If I use the set_uval and get_uval this 60MB will be copied for a while and it takes a lot of time and memory. I don't found a key to make a temporary copy using widgets. May be I am missing something.

Any ideas?

Reimar

--
R.Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

Subject: Re: Returning result from a widget program.
Posted by Vap User on Mon, 20 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

"R. Bauer" <r.bauer@fz-juelich.de> writes:

I haven't seen a reply, despite the fact that this article is a week
old on my server, so I'll do it. You asked the following at the end of
David Fanning's reply.

>
> I agree to you in not using common blocks in widgets in most cases.But how can I
> handle 60MB of data without a common block. If I use the set_uval and get_uval this
> 60MB will be copied for a while and it takes a lot of time and memory. I don't found a
> key to make a temporary copy using widgets. May be I am missing something.
>
> Any ideas?
>
> Reimar
>
>

I think the answer is to use a pointer to your large input data
thing, created with the /no_copy keyword set, in the 'info' structure,
that's the structure you pass in the Uvalu of the top level base of
the widget program. The secret is using /no_copy. By doing that you
are not making a copy of the data, but directly attaching the existing
memory to the new location. When you use /no_copy with Widget_Control,
you are undefining the local variable that had the data and attaching
the data instead to the Uvalue of the specified widget. When you use
/no_copy with PTR_NEW() you are undefining the local variable that had
the data before, and attaching it to the pointer. In neither case is a
copy of the data within the variable created.

I'll put some more comments in David's reply.

I hasten to add that you take the advice of David Fanning, et. al. to
heart. You'll thank yourself for doing it. AFter you get the knack of
writing widget programs with pointers and without common blocks,
you'll never go back to the other way of doing it.

William

>
> David Fanning wrote:
>
>> 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, ..., ...}

In your case, the info structure might look like

```
info = Ptr_new( { cancel:1, $
          dataPtr: Ptr_New( Large_Data_Thingie, /no_copy ), $
          ...,$
               ... } ) ; Possibly even with a '/no_copy here as well, although
                     ; this isn't as necessary.
```

Now the array 'Large_Data_Thingie' IS UNDEFINED within the current routine, but then you don't really need it here anyway.

```
>>     Widget_Control, tlb, Set_UValue=info, /No_Copy

        ; Note: the structure 'info' is undefined, but then, you don't really need it anyway.

>>     XManager, 'example', tlb   ; Modal widget blocks here.
>>
>>       ; User killed widget. Get data and return it.
>>

      Widget_Control, tlb, Get_Uvalue=info,/No_Copy ; Need this to get the Info structure back.

>>     data = *info.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
>>
>>
>

Reimars question moved to front of message.

>
>
>
>
> --

> R.Bauer
>
> Institut fuer Stratosphaerische Chemie (ICG-1)
> Forschungszentrum Juelich
> email: R.Bauer@fz-juelich.de
>
>
>

--
I don't speak for JPL, it doesn't speak for me.
Well, not all the time, at least.
William Daffer <vapuser@haifung.jpl.nasa.gov>