Subject: running an application from the IDL virtual machine
Posted by markb77 on Thu, 19 Nov 2015 17:58:19 GMT
View Forum Message <> Reply to Message

hi,

My IDL application is written as an object. When the object is created, the GUI of the application pops up, and the user may use the application until the application window is closed.

The application is started like this:

foo = obj_new('my_application')

My question is, how should I run this application from the VM?

If I create a script file containing the command above, and compile this into a SAV file, would that work?

What happens is this: everything starts OK. The object is created and the GUI of the application pops up. However, once the IDL VM finishes executing this line of code, it thinks that it has finished running the program. The VM closes, and my application is destroyed with the VM.

What is the solution? How can I keep the VM running, in an efficient way, until the user closes the application window?

thanks
Mark

Subject: Re: running an application from the IDL virtual machine
Posted by Jim Pendleton on Fri, 20 Nov 2015 00:52:13 GMT
View Forum Message <> Reply to Message

On Thursday, November 19, 2015 at 10:58:23 AM UTC-7, superchromix wrote:
> hi,
>
> My IDL application is written as an object. When the object is created, the GUI of the application pops up, and the user may use the application until the application window is closed.
>
> The application is started like this:
>
> foo = obj_new('my_application')
>
> My question is, how should I run this application from the VM?
>
> If I create a script file containing the command above, and compile this into a SAV file, would that work?
>

> What happens is this: everything starts OK. The object is created and the GUI of the application pops up. However, once the IDL VM finishes executing this line of code, it thinks that it has finished running the program. The VM closes, and my application is destroyed with the VM.
>
> What is the solution? How can I keep the VM running, in an efficient way, until the user closes the application window?
>
> thanks
> Mark

You will probably want your main widget event loop to be blocking if that's all your object is responsible for.

Subject: Re: running an application from the IDL virtual machine
Posted by markamatic on Fri, 20 Nov 2015 09:41:50 GMT
View Forum Message <> Reply to Message

On Friday, November 20, 2015 at 1:52:20 AM UTC+1, Jim P wrote:

> You will probably want your main widget event loop to be blocking if that's all your object is responsible for.

I don't understand what you mean here. Do you mean that the object initialization should never finish, and that the variable foo in the above example should not be returned, until the application closes?

The application itself is a graphics window. It can spawn further instances of itself if it needs to create new windows. The object initialization needs to finish smoothly, in order for things like that to work.

Subject: Re: running an application from the IDL virtual machine
Posted by Helder Marchetto on Fri, 20 Nov 2015 10:14:06 GMT
View Forum Message <> Reply to Message

On Friday, November 20, 2015 at 10:41:53 AM UTC+1, marka...@gmail.com wrote:
> On Friday, November 20, 2015 at 1:52:20 AM UTC+1, Jim P wrote:
>
>> You will probably want your main widget event loop to be blocking if that's all your object is responsible for.
>
> I don't understand what you mean here. Do you mean that the object initialization should never finish, and that the variable foo in the above example should not be returned, until the application closes?
>
> The application itself is a graphics window. It can spawn further instances of itself if it needs to

create new windows. The object initialization needs to finish smoothly, in order for things like that to work.

Hi,
the way I do this is to make an executable of a pro that calls the object.
This is how it looks like:

```
function myObj::init
;make your widget here or elsewhere within the object
;no need to block the widget
self.wBase = widget_base()
self.wlabel = widget_label(self.wBase, value='show something')
widget_control, self.wBase, /realize
xmanager, 'myObjWid', self.wBase, /no_block
print, 'wid obj initialized'
return,1
end

pro myObj__define
void ={myObj, wBase:0l, wLabel:0l}
end

pro runMyObj
obj = obj_new('myObj')
print, 'obj called, exit pro. Obj lives on.'
end
```

This does the trick for me.
Is this what you're looking for?

Cheers,
Helder

Subject: Re: running an application from the IDL virtual machine
Posted by Helder Marchetto on Fri, 20 Nov 2015 11:26:35 GMT
View Forum Message <> Reply to Message

On Friday, November 20, 2015 at 11:14:09 AM UTC+1, Helder wrote:
> On Friday, November 20, 2015 at 10:41:53 AM UTC+1, marka...@gmail.com wrote:
>> On Friday, November 20, 2015 at 1:52:20 AM UTC+1, Jim P wrote:
>>
>>> You will probably want your main widget event loop to be blocking if that's all your object is responsible for.
>>
>> I don't understand what you mean here. Do you mean that the object initialization should never finish, and that the variable foo in the above example should not be returned, until the application closes?

>>
>> The application itself is a graphics window. It can spawn further instances of itself if it needs to create new windows. The object initialization needs to finish smoothly, in order for things like that to work.
>
> Hi,
> the way I do this is to make an executable of a pro that calls the object.
> This is how it looks like:
>
> function myObj::init
> ;make your widget here or elsewhere within the object
> ;no need to block the widget
> self.wBase = widget_base()
> self.wlabel = widget_label(self.wBase, value='show something')
> widget_control, self.wBase, /realize
> xmanager, 'myObjWid', self.wBase, /no_block
> print, 'wid obj initialized'
> return,1
> end
>
> pro myObj__define
> void ={myObj, wBase:0l, wLabel:0l}
> end
>
> pro runMyObj
> obj = obj_new('myObj')
> print, 'obj called, exit pro. Obj lives on.'
> end
>
> This does the trick for me.
> Is this what you're looking for?
>
> Cheers,
> Helder

Hi,
just so to make sure FG also works, here is a minimal example. Once started, you can interact with the plot as usual.
What I always find strange, is that in my case the plot (widget_window) appears black until I run the mouse over the window...

But the rest works. Cheers, Helder

function myObj::init
;make your widget here or elsewhere within the object
;no need to block the widget
self.wBase = widget_base()
self.wWindow = widget_window(self.wBase)

```
widget_control, self.wBase, /realize
xmanager, 'myObjWid', self.wBase, /no_block
self.pp = plot(/test, current=self.wWindow)
print, 'wid obj initialized'
return,1
end

pro myObj__define
void ={myObj, wBase:0l, wWindow:0l, pp:obj_new()}
end

pro runMyObj
obj = obj_new('myObj')
print, 'obj called, exit pro. Obj lives on.'
end
```

Subject: Re: running an application from the IDL virtual machine
Posted by markb77 on Fri, 20 Nov 2015 13:39:19 GMT
View Forum Message <> Reply to Message

Thanks Helder - this works.

Also, it seems that each time I run the executable version of the application, it starts in a new IDL process, running on a different core of the CPU.

Therefore, running my application in the VM is an easy way to use multiple cores of the CPU, without requiring me to start up several IDL_IDLBridge processes, etc.

cheers
Mark
