Subject: call_external and IDL_ExecuteStr
Posted by Allan Whiteford on Wed, 21 Mar 2007 18:08:01 GMT
View Forum Message <> Reply to Message

Hi,

Am I allowed to do this:

++++++ callback.c ++++++

```
#include <stdio.h>
#include "export.h"

void callback(int argc,char *argv)
{
 int i;
 i=IDL_ExecuteStr("testme");
}
```

++++++ callback.pro ++++++

```
pro testme
 print,'hey'
end

pro callback
 a=call_external('callback.so','callback');
end
```

++++++ Compilation command ++++++

```
gcc -shared -o callback.so -I$IDL_DIR/external/ callback.c
```

----------?

Looking through the documentation I find:

Note: The functions documented in this chapter should only be
used when calling IDL from other programs their use in code
called by IDL via CALL_EXTERNAL or a system routine (LINKIMAGE,
Dynamically Loadable Module) is not supported and is certain to
corrupt and/or crash the IDL process.

(in help/online_help/Using_Callable_IDL.html)

So I guess that means I'm not allowed to. However, it does work:

IDL> callback

% Compiled module: CALLBACK.
hey
% Temporary variables are still checked out - cleaning up...
IDL>

To put the whole thing in context, I have an IDL application which calls a FORTRAN subroutine via call_external, this subroutine can take a few minutes to run and I'd like to display a progress bar in IDL. To do this I need the FORTRAN to be able to tell the IDL to update the progress bar. I'd like to be able to make a call in the FORTRAN to an external IDL subroutine without having to return control to IDL and then re-enter the FORTRAN. I guess signals and RPC stuff would be something else to look at but the above solution is really neat (if it's safe!).

Is the above note simply scaremongering to reduce support requests and bug reports or will some really bad happen with the example code? Is there a better way to achieve the above which doesn't conflict with the documentation?

I've found:

http://cow.physics.wisc.edu/~craigm/idl/archive/msg00589.htm l

which I guess is the reason for the warning going in to the documentation. However, if I don't use IDL_ExecuteStr to allocate memory or anything else I feel it will probably be fairly safe. Do people have any experience and/or thoughts?

I've also found:

http://cow.physics.wisc.edu/~craigm/idl/archive/msg05117.htm l

where a number of suggestions are made. One, by JD Smith is suggesting possibly using the same solution as my independent example above but in a linkimage context. However, I think this clashes with the documentation note as well. It's often a good bet to assume JD Smith knows more than the documentation but I'd like a bit more reassurance.

Any reassurance, warnings or alternative solutions welcome and very much appreciated.

Thanks,

Allan

Subject: Re: call_external and IDL_ExecuteStr
Posted by codepod on Thu, 22 Mar 2007 14:09:48 GMT
View Forum Message <> Reply to Message

Allan:

While you can often get away with using IDL_ExecuteStr() from a call_external()/Linkimage/DLM routine, it's not really supported by IDL. IDL_ExecuteStr() is intended for use within the context of Callable IDL and assumes that you are calling the interpreter at the $MAIN$ level.

When you use IDL_ExectueStr() from a system called routine, any variable allocation or error condition can cause side effects and potentially exit IDL. Most cases the issues are not directly noticeable, but leave IDL in an unstable state and result in unexpected behavior over the period of execution.

- CP

On Mar 21, 12:08 pm, Allan Whiteford <allan-remove-th...@-and-this.phys-dot-strath.ac.uk> wrote:
> Hi,
>
> Am I allowed to do this:
>
> ++++++ callback.c ++++++
>
> #include <stdio.h>
> #include "export.h"
>
> void callback(int argc,char *argv)
> {
> int i;
> i=IDL_ExecuteStr("testme");
>
> }
>
> ++++++ callback.pro ++++++
>
> pro testme
> print,'hey'
> end
>
> pro callback
> a=call_external('callback.so','callback');
> end
>

> ++++++ Compilation command ++++++
>
> gcc -shared -o callback.so -I$IDL_DIR/external/ callback.c
>
> ---------------------------------?
>
> Looking through the documentation I find:
>
> Note: The functions documented in this chapter should only be
> used when calling IDL from other programs their use in code
> called by IDL via CALL_EXTERNAL or a system routine (LINKIMAGE,
> Dynamically Loadable Module) is not supported and is certain to
> corrupt and/or crash the IDL process.
>
> (in help/online_help/Using_Callable_IDL.html)
>
> So I guess that means I'm not allowed to. However, it does work:
>
> IDL> callback
> % Compiled module: CALLBACK.
> hey
> % Temporary variables are still checked out - cleaning up...
> IDL>
>
> To put the whole thing in context, I have an IDL application
> which calls a FORTRAN subroutine via call_external, this
> subroutine can take a few minutes to run and I'd like to display
> a progress bar in IDL. To do this I need the FORTRAN to be able
> to tell the IDL to update the progress bar. I'd like to be able
> to make a call in the FORTRAN to an external IDL subroutine
> without having to return control to IDL and then re-enter the
> FORTRAN. I guess signals and RPC stuff would be something else
> to look at but the above solution is really neat (if it's
> safe!).
>
> Is the above note simply scaremongering to reduce support
> requests and bug reports or will some really bad happen with the
> example code? Is there a better way to achieve the above which
> doesn't conflict with the documentation?
>
> I've found:
>
> http://cow.physics.wisc.edu/~craigm/idl/archive/msg00589.htm l
>
> which I guess is the reason for the warning going in to the
> documentation. However, if I don't use IDL_ExecuteStr to
> allocate memory or anything else I feel it will probably be
> fairly safe. Do people have any experience and/or thoughts?

>
> I've also found:
>
> http://cow.physics.wisc.edu/~craigm/idl/archive/msg05117.htm l
>
> where a number of suggestions are made. One, by JD Smith is
> suggesting possibly using the same solution as my independent
> example above but in a linkimage context. However, I think this
> clashes with the documentation note as well. It's often a good
> bet to assume JD Smith knows more than the documentation but I'd
> like a bit more reassurance.
>
> Any reassurance, warnings or alternative solutions welcome and
> very much appreciated.
>
> Thanks,
>
> Allan
