Subject: Question for RSI: Call_external/dlms vs Callable IDL...
Posted by steinhh on Wed, 21 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

It's beginning to dawn on me that I may have misunderstood something fundamental about the nature of call_external code vs Callable IDL. This question is mostly directed towards the friendly people at RSI who follow this newsgroup closely, but if anyone else have information on this subject, please speak up.

Is it correct, as I now suspect, that none of the routines described in the "Callable IDL" section of the External Development Guide are supposed to work from within call_external/dlm routines?

I.e., is it supposed to be impossible to use calls like IDL_ExecuteStr("CALL_PROCEDURE,'myprog',F") inside an added system routine?

If so, then I *strongly* suggest that you provide at least two extra functions in the next version, e.g.:

```
vptr = IDL_Call_Function("myfunc",argc,argv [,argk]);
IDL_Call_Procedure("myproc",argc,argv [,argk]);
```

or possibly more generally:

```
vptr = IDL_CallSysFunc("CALL_FUNCTION",argc,argv [,argk]);
IDL_CallSysProc("CALL_PROCEDURE",argc,argv [,argk]);
```

The reason I'm asking is that I'm trying to implement the MINPACK minimization routines (in Fortran) as system routines in dlms, and for that purpose I need to be able to call user-written procedures with input/output parameters.

Until yesterday, I was to be under the impression that an added system routine had a local variable space of it's own (since IDL_M_NAMED_GENERIC messages do report errors to occur inside a named routine just like the MESSAGE procedure does inside normal procedures), and that (named!) local variables could safely be created/retrieved within this scope with IDL_FindNamedVariable(), and referred to in execute statements like "CALL_PROCEDURE,'myprog',F" (communicating the contents of local variable "F" to and from routine "myprog").

If that's the way it's supposed to work, I have a bug report. Consider the files idlmfun.pro, idlmder.c and

idlmder.dlm below for an absolutely *minimal* example. As it is, you may execute the function idlmder both with and without a parameter, with no problem. Note, however, that the variable FF referred to in the execute string is created at the $MAIN$ level.

Now, delete the IDL_ExecuteStr("help") call, and recompile into a dlm. Calling idlmder with no argument still works, but when given a parameter it crashes:

IDL> idlmder ;; Ok
IDL> idlmder,n
Segmentation fault

This is on { alpha OSF unix 5.2 Oct 30 1998}, btw.

If this is not supposed to work, well, then the call_external/dlm mechanism is a lot less useful than it could be (crippled, I'd say), and you definitely ought to spend the (quite small) effort to tell us how to interface with the internals of IDL to avoid it..

Those who have gone to great lengths to understand dlms in the first place are surely capable of understanding how to call the functions I've sketched above, with almost no further explanation. Also, having functions like this would be a benefit anyway, since it'll allow dlms to execute IDL statements (through procedures/functions) without the compilation overhead that's associated with every IDL_ExecuteStr call (even when repeating the same statement a zillion times).

Regards,

Stein Vidar

idlmfun.pro----------cut
```
PRO idlmfun,pder
  pder = dblarr(10)
END
```
----------cut

idlmder.c----------cut
```
#include "stdio.h"
#include "export.h"

void IDLMDER(int argc, IDL_VPTR argv[])
{
```

```
  IDL_ExecuteStr("CALL_PROCEDURE,'idlmfun',FF");
  IDL_ExecuteStr("help");/* Delete this & die! */
}

int IDL_Load(void)
{
  static IDL_SYSFUN_DEF proc_def[] = {
    {(IDL_FUN_RET) IDLMDER,"IDLMDER",0,1}
    };
  return  IDL_AddSystemRoutine(proc_def,FALSE,IDL_CARRAY_ELTS(proc_def ));
}
-----------------------------------------------------cut

idlmder.dlm-----------------------------------------cut
MODULE IDLMDER
DESCRIPTION Subroutines: LMDER
BUILD_DATE 21 Apr 1999
SOURCE S. V. H. Haugan
PROCEDURE IDLMDER 0 1
end-------------------------------------------------cut
```

Subject: Re: question
Posted by davidf on Thu, 09 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Ophir Maor (pro_mao@hmi.de) writes:

> 1. if i have a window and inside i 'm getting some data from the user,
> and then i want put two buttons for o.k. and cancel... how can i do it,
> meaning when presssing o.k. the data will be transfered to the callback
> function of the button 'o.k.'

You want to create what is called a "modal dialog widget".
When the user hits the OK button, collect the information
from the form and store it at a global pointer location. Then
destroy the widget, which releases the modal block (sometimes
just a command line block), allowing you to retrieve the
information from the pointer and return it to the user.

You can read Chapter 13 of my book for all the gritty
details, or just grab some example code off my web page
if you want to learn by doing. You could start with
the GetImage program, for example:

  http://www.dfanning.com/programs/getimage.pro

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155