Subject: Re: CALL_EXTERNAL puzzle (still) ?
Posted by Armand J. L. Jongen on Fri, 04 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

After trying to follow this discussion, there are a few comments which
might be helpfull to you:

>> However, why can't I pass a pointer?
>
> Good question - I would have liked to be able to exploit
> pointers inside external code, but RSI has (at the moment)
> specifically forbidden that:
>
> Direct access to pointer and object reference heap
> variables (types IDL_TYP_PTR and IDL_TYP_OBJREF,
> respectively) is not allowed.
>
> (External dev. guide, "Heap Variables")
>
> So, although the pointer is passed, you cannot do anything
> with it. Please, RSI? Why not some function like
>
> IDL_VPTR IDL_GetHeapVariable(HVID)

There is a way! At least I think :-))
When developing an intermediate DLL to control a framegrabber I ran into
the following problems:

Problem:

The ID of the framegrabber was of an unknown structure (protected by the
manufacturer for future development..... sounds like RSI:-)), but I
wanted to "know" inside IDL..... and pass the pointer to that unknown
structure back into DLL for other procedures.....

Solution:

I defined a pointer in a structure

info={DevPointer : Ptr_New()}

Beacause you cannot change it in the DLL (as RSI told us) I thought I
might just as well let it be returned by the Dll!

Then the C-code looked a bit like this:

```
Unknown_ID WINAPI FG_OpenDevice( LONG lArgc, LPVOID lpvArgv)
{
```

```
 LPLONG    lplArgv = NULL;
 CHAR    szMsg[256];

 lplArgv    = (LPLONG)lpvArgv;

    /* Try to open the device */

if (OpenDevice(&DevicePointer) != 0) {
 return -200;
}
 return DevicePointer;
}
```

where the Unknown_ID was that protected structure. The IDL code to get
the DevPointer was like:

```
Ptr_Free, info.DevPointer
info.DevPointer = Ptr_New(Call_External('idl_fg.dll', 'FG_OpenDevice'))
```

When wanting to pass the DevPointer as a pointer to the DLL (as I didn't
know the structure of the Unknown_ID this was the only way) I passed the
dereferenced pointer !!!!by reference!!!! like

```
lResult = Call_External('idl_fg.dll', 'FG_SomeProcedure',
(*(info.DevPointer)), $
  Value=[0b])
```

Then IDL passes in fact the pointer to the DLL, just what I wanted!!

>> And if I want to pass a pointer, and
>> print the value of another pointer just before the CALL_EXTERNAL, why is the
>> wrong one passed?
>
> These are how your "bad luck" arose, apparently. Probably the
> print statement caused the text to be stored right after the
> space where the first pointer was located? Expect *no*
> consistency from platform to platform, or IDL version to IDL
> version!

When developing the application where I used the above mentioned
technique, at one point everything seemed to work perfectly! Foolproof
so to say. The only thing was that when I started the application for
the second time in one IDL session, it crashed. Restarting IDL made it
work again! After three nights of how, why, hell etc... I found out that
some akward ID of a widget! was used for the ID of a framebuffer and
fortunately (or unfortunately:-(((whichever way you look at it) these
where the same when starting a fresh IDL session!!! The message: if

things work, it is not always so that things are running the way you
think they do!

Hopefully all this is not to much (crap), and maybe a bit usefull.

A last note (It's almost weekend:-):

When passing (BOLD) structures (BOLD) by reference, IDL does not pass
the normal pointer to the DLL. In fact it makes a copy of the structure
and passes the pointer to that copy into the operand. Therefore only the
copy is changed and not the original! A workaround is to make a copy
yourself before passing, and assigning the (altered) copy back to the
structure after your DLL-call. Example:

```
tmp_RefVolt = (*ptr).RefVolt ; Here I expect a value

Result = Call_External('idl_fg.dll', 'FG_GetRefVoltage', $
       (*ptr).InputSource, tmp_RefVolt, Value=[1b,0b])

(*ptr).RefVolt = tmp_RefVolt
```

In these cases it is again better to let the DLL return the wanted
values as shown above. When passing dereferenced pointers by refernce
thing work as expected so e.g.

```
info={image:Ptr_New()}

Ptr_Free, info.image
info.image=Ptr_New(bytarr(256,256))

lResult = Call_External('idl_fg.dll', 'FG_GetFrame', $
  (*(info.Image)), Value=[0b])
```

fills the array with the captured image (provided the DLL is written
properly:-))

Enough for now, have a nice weekend!

Cheers,

--
 ** ************
Armand J.L. Jongen   Academic Medical Centre
                                  Laser Centre
Phone   +31-20-5667419   \\||||//   Meibergdreef 9
Fax   +31-20-6975594 | ~ ~ | 1105 AZ Amsterdam
E-mail a.j.jongen@amc.uva.nl   [| o o |]   The Netherlands
 ****************************o00o***(__)***o00o************* ************