Subject: Re: how to pass array from dll into IDL?
Posted by nmw on Fri, 06 Jul 2001 10:24:52 GMT
View Forum Message <> Reply to Message

In article <fbb56367.0107050653.2730fd68@posting.google.com>, maxschautzer@my-deja.com writes:
> Hi ,
>
> I need to pass a bytearray from the dll to IDL (5.2).
> I am writing a wrapper for a framegrabber which uses the dll functions to aquire
> the single frames(monochrome).
>
> Then I try to pass the pointer address to idl, then this address seems not to work.
> Here my example:
>
> in dll:
>
> IDL_LONG TestFunction(int argc, void* argv[])
> {
> IDL_LONG *answer;
> BYTE b=255; //BYTE is UCHAR
> BYTE *pb;
> answer = (IDL_LONG *)argv[0];
> pb=&b;
> *answer = (IDL_LONG)pb;
> return(0L); //Return zero if all okay
> }
>
> in IDL:
>
> a=10L
> err=Call_External('aquire.dll','TestFunction',a)
> print,a
> IDL>10415668
>
> This address seems not to be the address for b in dll. I tested this with:
> var = EXTPROC_DEREF(addr,BYTE=1)
> print,var
> IDL>94 not 255
> -> EXTPROC_DEREF comes from http://www.rlkling.com/
>
> So her are my questions: How can I access these different address spaces?
> How can I test this address in IDL?
> Is there any possibility to transfer whole arrays(frames)?
> And if, how fast can I get these arrays(25ms)?
>
> Thanks
>

> Max

Personally, I'd forget trying to use call_external for this. Using that
method you can only fill in the contents of variables which are created
in IDL and passed to your code.

You're much better off going down the DLM route. You can then create IDL
variables in your code and pass them back to IDL. It's a steeper learning
curve but will pay off in the end. I'd advise getting Ronn's book on
"Calling C from IDL". Alternatively you could learn it the hard way from
the IDL on-line docs ...

Here's an example to get you started and to demonstrate just how simple it
really is ;-) This DLM creates a byte array, fills it with values and passes
it back to IDL. Copy the C code into a file data.c, and the DLM into a file
data.dlm. Then compile the code into a shared object, on UNIX this is simply

```
 cc -shared -o data.so data.c -I<path_to_IDL>/external
```

I don't know how to do this under Windows as I don't use that OS, hopefully
someone in the group will be able to help. Failing that, there are
instructions in the IDL online help (edg.hlp, section "Adding System
Routines").

To use it run IDL:

```
ion:nmw 565$ idl_5.2
IDL Version 5.2.1 (IRIX mipseb). (c) 1999, Research Systems, Inc.
Installation number: 3437-9.
Licensed for use by: Radio and Space Plasma Physics Group

IDL> a=get_data()
% Loaded DLM: DATA.
IDL> print,a
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27
28  29
  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55
 56  57  58
  ...
```

Good luck.

--

Nigel Wade, System Administrator, Space Plasma Physics Group,
          University of Leicester, Leicester, LE1 7RH, UK
E-mail :   nmw@ion.le.ac.uk

Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555

Here are the source files:

data.c:

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

#include "export.h"

#define DATALEN 250

IDL_VPTR get_data(int argc, IDL_VPTR argv[], char *argk)
{

   IDL_VPTR array;
   IDL_LONG dims[IDL_MAX_ARRAY_DIM];

   IDL_VPTR result;

   unsigned char *data;
   int i;

   /*
    * get the data
    */
   data = malloc(DATALEN);
   if ( data == NULL )
 IDL_MESSAGE( IDL_M_NAMED_GENERIC, IDL_MSG_RET, strerror(errno));

   for ( i = 0; i < DATALEN; i++ )
 data[i] = i + 1;

   /*
    * create the return data array
    */
   dims[0] = DATALEN;

   array = IDL_ImportArray(1, dims, IDL_TYP_BYTE, data, NULL, NULL);

   return array;
}
```

```
int IDL_Load(void) {

  static IDL_SYSFUN_DEF function_addr[] = {
    { get_data, "GET_DATA", 0, 0, 0},
  };

  return IDL_AddSystemRoutine(function_addr, TRUE, 1);
}
```

data.dlm:

```
MODULE data
DESCRIPTION Create a simple data array.
VERSION 1.0
SOURCE Nigel Wade, Leicester University.
BUILD_DATE July 6, 2001
FUNCTION get_data  0 0
```