
Subject: Re: Passing Structures with Pointers with Call_External
Posted by [MajorSetback](#) on Wed, 11 Aug 2004 13:53:25 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Peter Mason" <drone@spam.com> wrote in message
news:<vDcSc.266\$aA.11145@news.optus.net.au>...

> PeterOut wrote:

> <...>

>>

> temp={Rows:long(numrows),Columns:long(numcolumns),Data:ftlar r(numrows,numcol
> umns)}

> <...>

>> The C code is as follows.

>> typedef struct FloatPlane_Struct

>> {

>> long Rows;

>> long Columns;

>> float **Data;

>> } FloatPlane;

> <...>

>> If I add

>> fprintf(stderr, "fppPlanes->Data[0]=%d\n", fppPlanes->Data[0]);

>> idlde crashes, presumably due to a memory write error in the C code.

>> Is there any way to stop idlde crashing under such circumstances?

>>

>> My main question is this. Is there a way to retrieve the IDL variable

>> Planes[i].Data within CFunction_cw?

>

>

> The problem here is that IDL isn't creating the structure quite as you

> expect. There isn't that level of indirection with DATA. Your C-side

> structure should look something like this:

> typedef struct FloatPlane_Struct {

> int Rows;

> int Columns;

> float Data[n];

> } FloatPlane;

> Where the "n" in "Data[n]" is equal to numrows*numcolumns in your IDL-side

> structure creation statement.

> I think this means that you need a different approach as a C-side structure

> definition is fixed at compile time ("n" must be a constant).

>

> You might be wondering about changing your structure definition to use an

> IDL "pointer" for the array? Don't even try it. The value of an IDL

> pointer is like some handle index thing and bears no relation to an actual

> memory address. It's meaningless to external code.

>

> Personally, I'd suggest abandoning the use of a structure and coding a DLM

> instead of CALL_EXTERNAL here. CALL_EXTERNAL is quick and easy but
> sometimes it's worth going that extra distance. In a DLM you would be able
> to pull out the dimensions of your DATA array (now 3-dimensional for the
> frames). Also, the IDL-side work would probably be more efficient with a
> straightforward array instead of arrays embedded in structures.
> Alternatively, stick to CALL_EXTERNAL and pass your C function two
> parameters: DATA and SIZE(DATA).
>
> Peter Mason

Hi Peter,

Thanks very much for the reply. I decided to do something along the lines of the last thing you mentioned. I made a 3D volume with IDL thus.

```
Volume=fltarr(nPlanes,nRows,nCols)
```

I then filled the volume with the plane data and pass it thus.

```
Result=Call_External('SharedLibrary.so','CFunction_cw',NumPlanes,NumRows,NumCols,Volume,/unload)
```

In the C function, I then have a float pointer that reads the array thus.

```
float *fpData=((float *)(argv[3]));
```

Hence I have a pointer to the data, in 1D, that can be upwrapped using the volumetric dimensions.

Thanks again for your help,
Peter.
