Subject: Re: Passing Structures with Pointers with Call_External
Posted by Bob[2] on Wed, 11 Aug 2004 12:40:57 GMT
View Forum Message <> Reply to Message

Peter, you might try changing the C-side to accept a struct of many Planes, then use call_external with a single argument (the name of the struct) and set /autoglue. E.g.,

```
function MakePlanes, numrows, numcolumns, sNumberOfPlanes
 ; put it together.  Must be named FloatPlane_Struct.
 return,a_planes_struct
end
```

On the C-side,

```
typedef struct FloatPlane_Struct
{
 /* as before
}

IDL_INT work( struct FloatPlane_Struct *a )
{
 ; Do your work with a->Rows[j] and so on
 ; then return to IDL.

 return,0;
}
```

And from the IDL-side again, e.g.

```
my_planes = MakePlanes(33,12,5)

work_result = call_external('SharedLibrary.so','work',my_planes, $
 /AUTO_GLUE,/I_VALUE)
```

Now, as I have set up my example, 'work' does computations on the structure 'my_planes' and maybe changes values in it etc., then returns an IDL_INT when finished. You can make the return type of 'work' to be something else, like a struct, and instead of /I_VALUE you set RETURN_TYPE=8.

Have fun working with it.

I am not sure why your other stuff is crashing idlde, though.

> I declare an array of structures thus.
> function MakeFloatPlane,numrows,numcolumns

```
>       temp={Rows:long(numrows),Columns:long(numcolumns),Data:fltar
r(numrows,numcolumns)}
>      return,temp
>  end
>
>  Planes=replicate(MakeFloatPlane(nY1,nX1),sNumberOfPlanes)
>
>  I then pass the array to C code through Call_External thus.
>   Result=Call_External('SharedLibrary.so','CFunction_cw',sNumb erOfPlanes,Planes,/unload)
>
>  The C code is as follows.
>  typedef struct FloatPlane_Struct
>  {
>          long    Rows;
>          long    Columns;
>          float   **Data;
>  } FloatPlane;
>
>  extern "C" long CFunction_cw(int argc, void *argv[])
>  {
>          long    lNumberOfPlanes;
>          FloatPlane *fppPlanes;
>          float   *fpData;
>
>          lNumberOfPlanes=*((long *)(argv[0]));
>          fppPlanes=((FloatPlane *)(argv[1]));


>  However, I cannot interpret the result I get for fppPlanes->Data.
>
>  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?
```