Subject: Re: CALL_EXTERNAL and strarr()
Posted by Nigel Wade on Fri, 10 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Richard Tyc wrote:
>
> I need to call an external C function (unix/shared lib method) which
> will collect a string array of different size strings. I have been
> using call_external with success on returning other types (floats,
> strings etc) but this seems more challenging. The problem being I don't
> know how large each string will be in the string array so it needs to be
> dynamically allocated.
>
> I know the following can pass IN a string array and return a single
> string:
>
> replyStr = CALL_EXTERNAL('IDLquery_call.so','GetPatientList',strarr,$
> N_ELEMENTS(strarr), /S_VALUE)
>
> the key point in the C function being to make the returned string static
> to avoid memory leaks.
>
> Anyone care to point out how I might be able to retrieve a string array.
>
> I was thinking about IDL_ImportArray or IDL_ImportNamedArray to create
> an IDL Array.
>
> so lets say in my C function I have
>
> char *list[100] ; or should it be IDL_STRING *list100 ; ??????
>
> and the function has gone off an filled this list with various sized
> array strings.
> (eg. list[0] = strdup("5char") list[1] = strdup("06char") where strdup
> is using malloc implicitly)
>
> Assuming I create an IDL array from this data, what mechanism do I have
> of returning this data in the call_external function.
>
> Thanks in Advance
> Rich

First off, I'm not really a CALL_EXTERNAL user, I generally use
LINKIMAGE
or DLMs, but here goes anyway.

From the call structure you've outlined above, it looks as though you
know when you call the routine how many strings you are going to return

and pass in an array of strings. Is that right?

If so, within the external code can you not just step through the string array and change the strings using IDL_StrStore? Something like:

```
IDL_ARRAY string_array;
IDL_STRING *strings;

string_array = argv[1]->value.arr;
strings = (IDL_STRING *)string_array->data;

for ( i=0; i<string_array->n_elts; i++ ) {
   IDL_StrStore(&strings[i], list[i]);

   /* if you don't need list any longer, free the string
      IDL_StrStore creates its own copy */
   free(list[i]);
}
```

Now, I'm not saying this will work. It's just a suggestion as something to try. It may cause IDL to die horribly, or on Windows give the infamous
BSOD, when you try to store the strings into the string array since the array was created by IDL. If I were using LINKIMAGE/DLM then I would create the string array in the C code and pass it back to IDL.

--

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