Subject: Re: Problems with IDL call_external to C shared object
Posted by John-David T. Smith on Thu, 12 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Streun Andreas wrote:
>
> Hello -
>
> who is experienced in running IDL with C shared objects?
>
> I'm trying to make an IDL GUI for a rather complex C batch
> program. The effects are rather strange: sometimes it works
> perfectly, but mostly it doesn't: suddenly on the C-side strange
> and wrong number appear in calculations leading to crashes. It seems
> like something is initialized or dereferenced in a wrong way.
> However the behaviour is determinsitic: a small change in the IDL
> program like declaring a new variable anywhere causes the crash,
> after undoing the change it works well again. Maybe a memory conflict ?
>
> The C-program alone in batch mode runs reliably. It does a lot of
> mallocs but never frees any memory (because it is batch).
> IDL communicates via the CALL_EXTERNAL function.
> I'm rather sure that I have checked the variables on both sides of the
> fence are really of same type. (However I'm a poor C-programmer...)
> I'm using IDL 5.3 on a Linux system and the GNU C-compiler.
>
> Now the questions:
>
> - Is it possible that IDL overwrites or frees memory allocated by the C
> shared object ? Is there a general way to prevent it from doing so ?
>
> - What is the effect of the /CDECL keyword to CALL_EXTERNAL ?
> I tried with and without but no success.
>
> - Is it possible that the C program "forgets" something between
> the IDL CALL_EXTERNALs ?
>
> (important:)
> - Is there an opinion whether this problem can be solved in principle
> and within finite time ?!
>
> (has nothing to do with the problem but I would like to know:)
> - How can I return an array via CALL_EXTERNAL or have I always
> to loop over calls returning scalars ? The EZCA library (channel
> access to EPICS control system) manages to return arrays, but I
> couldn't figure out how.
>
> Thanks for any help.

The best way to use call_external I've found is to allocate all arrays, variables, and strings on the IDL side and directly manipulate them within the C program. Most variable types do *not* map directly between IDL and C. Did you take a good look at $IDL_DIR/external/call_external/C/, which contains lots of (small) examples? Also see the "external.h" header for lots of info.

Another bit of confusion: IDL simply calls the function directly from the shared library specified... the function is not at all linked in (other than existing in a shared program stack), and variables will not be preserved through successive function calls (unless they are declared static or global).

An example of passing an array as a variable:

```
IDL_LONG showarray(int argc, void *argv[]) {
  float *arr;
  IDL_MEMINT *n_elem,i;
  arr=(float *) argv[0];
  n_elem=(IDL_MEMINT *) argv[1];
  printf("%d\n",*n_elem);
  for(i=0;i<*n_elem;i++)
 printf("%d: %f\n",i,arr[i]); /* Don't printf, it's not nice! */
  return 1;
}
```

which would be called via, e.g.:

```
IDL> ret=call_external('mylib.so','showarray',findgen(10),10)
```

Presumably if your code allocates it's own memory without cleaning up after itself, it will be rather unstable. Unless you need to return arrays of dynamic size/type, the originate-all-data-in-IDL method will much simplify your life.

Good luck,

JD

--
J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |