Subject: Re: Call_external fails on Sun/Solaries
Posted by afl on Mon, 30 Jan 1995 22:57:20 GMT
View Forum Message <> Reply to Message

In article <3gjm3d$7o4@hammer.msfc.nasa.gov>, chaganti@ssl.msfc.nasa.gov writes:
|> Dear IDL GURUS,
|>
|> We have code written for VMS/FORTRAN. IDL code uses the CALL_EXTERNAL.
|> Now we are posting to Sun/Solaris/fortran.
|>
|> It seems the variable passing to Call_external on Sun is quite different
|> from VMS.
|>
|> Does any one tried to port the call_external code from VMS to Sun?
|>
|> Could some one give me an example? The examples in dynamic_library
|> pass varibale differently for VMS and SUN.
|>
|> Thanks
|> Kris Chaganit
|> chaganti@ssl.msfc.nasa.gov
|> exit

Before you continue, call RSI technical support.
I was told (by them) that CALL_EXTERNAL does not work
with version 3.6.X of IDL because of some bug. I would
call them before continuing work on this topic.

--

Andrew F. Loughe email: afl@cdc.noaa.gov
University of Colorado, CIRES voice: (303) 492-0707
Campus Box 449 fax: (303) 497-7013
Boulder, CO 80309-0449 USA

Subject: Re: Call_external fails on Sun/Solaries
Posted by rivers on Tue, 31 Jan 1995 01:53:36 GMT
View Forum Message <> Reply to Message

In article <3gjm3d$7o4@hammer.msfc.nasa.gov>, chaganti@ssl.msfc.nasa.gov writes:
> Dear IDL GURUS,
>
> We have code written for VMS/FORTRAN. IDL code uses the CALL_EXTERNAL.
> Now we are posting to Sun/Solaris/fortran.
>
> It seems the variable passing to Call_external on Sun is quite different

> from VMS.
>
> Does any one tried to port the call_external code from VMS to Sun?
>
> Could some one give me an example? The examples in dynamic_library
> pass varibale differently for VMS and SUN.
>

Indeed CALL_EXTERNAL on VMS and Unix pass arguments very diferently.

IDL under VMS uses the simple mechanism of passing each variable either by reference (everything but strings) or by descriptor (strings). Arguments can be passed by value by explicitly setting a flag in CALL_EXTERNAL.

Under Unix CALL_EXTERNAL passes arguments by the (argc, argv) mechanism. argc is a count of the arguments and argv is a vector of addresses of the arguments. The address for a string is the address of a descriptor.

PV-WAVE has yet another difference. It passes strings by reference, not by descriptor.

The IDL quirks for various operating systems is all documented in IDL_DIR/misc/dynamic_link

It is possible to write C code which works with all of them. Here is an exerpt from one of my programs. Note that it defines different macros depending upon the number of arguments your routine will be called with. This may not be the most elegant way to do, but it is the only way I could think of. I think Chris Jacobsen may have a more general method. My code has been tested with IDL and PV-WAVE on Unix and IDL on VMS. PV-WAVE on VMS has not been tested.

Mark Rivers (312) 702-2279 (office)
CARS (312) 702-9951 (secretary)
Univ. of Chicago (312) 702-5454 (FAX)
5640 S. Ellis Ave. (708) 922-0499 (home)
Chicago, IL 60637 rivers@cars3.uchicago.edu (Internet)

********** **********

```
/* The following macros allow for the differences in the way PV-WAVE and
*  IDL pass character strings. PV-WAVE passes the address of the address
*  of the string in argp[]. IDL passes the address of a string descriptor
*  structure, which contains the address of the string as one of its elements */
#ifdef IDL
typedef struct {
   unsigned short length;
   short type;
```

```
   char *address;
} IDL_STR_DESCR;
#define STRARG IDL_STR_DESCR
#define STRADDR(s) s->address
#define STRFIXLEN(s) *(s->address + strlen(s->address)) = (char) 32
static char* str_array_addr[100];
#define BUILD_STR_ARRAY(len, s) for (i=0; i<len; i++) str_array_addr[i]=s[i].address
#define STR_ARRAY_ADDR(s) str_array_addr

#else /*(PV-WAVE)*/
#define STRARG char*
#define STRADDR(s) *s
#define STRFIXLEN(s) (*s)[strlen(*s)] = (char) 32
#define BUILD_STR_ARRAY(len, s)
#define STR_ARRAY_ADDR(s) s
#endif

/* The following macros allow this source file to be used with
   PV-WAVE and IDL on both Unix and VMS platforms. The difference
   is that on VMS platforms arguments are passed directly
   (by reference), while on Unix they are passed by the (argc, argp)
   mechanism. These macros also simplify the code for each routine. */
# if defined (VMS)
#   define WAVE_HEADER0(ftype, fname)\
      ftype fname() {
#   define WAVE_HEADER1(ftype, fname, type1, arg1)\
      ftype fname(type1 *arg1) {
#   define WAVE_HEADER2(ftype, fname, type1, arg1, type2, arg2)\
      ftype fname(type1 *arg1, type2 *arg2) {
#   define WAVE_HEADER3(ftype, fname, type1, arg1, type2, arg2, type3, arg3)\
      ftype fname(type1 *arg1, type2 *arg2, type3 *arg3) {
#   define WAVE_HEADER4(ftype, fname, type1, arg1, type2, arg2, type3, arg3, type4, arg4)\
      ftype fname(type1 *arg1, type2 *arg2, type3 *arg3, type4 *arg4) {
# else
#   define WAVE_HEADER0(ftype, fname)\
   ftype fname(argc, argp)\
    int argc;\
    void *argp[];\
    {
#   define WAVE_HEADER1(ftype, fname, type1, arg1)\
   ftype fname(argc, argp)\
    int argc;\
    void *argp[];\
    {\
    type1 *arg1 = (type1 *) argp[0];
#   define WAVE_HEADER2(ftype, fname, type1, arg1, type2, arg2)\
   ftype fname(argc, argp)\
    int argc;\
```

```
    void *argp[];\
    {\
    type1 *arg1 = (type1 *) argp[0];\
    type2 *arg2 = (type2 *) argp[1];
#   define WAVE_HEADER3(ftype, fname, type1, arg1, type2, arg2, type3, arg3)\
   ftype fname(argc, argp)\
    int argc;\
    void *argp[];\
    {\
    type1 *arg1 = (type1 *) argp[0];\
    type2 *arg2 = (type2 *) argp[1];\
    type3 *arg3 = (type3 *) argp[2];
#   define WAVE_HEADER4(ftype, fname, type1, arg1, type2, arg2, type3, arg3, type4, arg4)\
   ftype fname(argc, argp)\
    int argc;\
    void *argp[];\
    {\
    type1 *arg1 = (type1 *) argp[0];\
    type2 *arg2 = (type2 *) argp[1];\
    type3 *arg3 = (type3 *) argp[2];\
    type4 *arg4 = (type4 *) argp[3];
#endif

/* Here is an example of using these macros */

/*************************************************
  set ca_pend_event timeout
*************************************************/
WAVE_HEADER1(void, CaWavePendEventTime, double, d)
        CA.PEND_EVENT_TIME = *d;
        if (CA.devprflag > 0) fprintf(stderr,"Set pend_event_time=%f\n",CA.PEND_EVENT_TIME);
}
```