Subject: Re: Data passing & CALL_EXT
Posted by rivers on Fri, 21 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <5ekl3mmnp1@hammer.msfc.nasa.gov>, mallozzi@ssl.msfc.nasa.gov writes:
> Hi,
>
> Is it possible to pass the following data structures from C code back
> to IDL via CALL_EXTERNAL:
>
> 1. An array of strings?
> 2. A C structure, containing int, float, char
> 3. A C union
>
> I got my C code to pass back a single string, but could not do an
> array of strings. Really I would like to pass a structure back to
> IDL. Does anyone have any examples of this? The IDL examples treat
> only the simplest cases.
> PS I am using OpenVMS :-(

The answer is yes, you can do all of the above.

1) Arrays of strings work. However, note that IDL and PV-WAVE pass strings to CALL_EXTERNAL differently. IDL passes strings by descriptor, PV-WAVE passes them by reference. It is sometimes more convenient to convert IDL strings to byte arrays before calling your C code and then convert them back to IDL strings after the return from CALL_EXTERNAL. Byte arrays are handled the same in IDL and PV-WAVE.

2) IDL structures are just like C structures. You can pass them back and forth with no problem. In my experience IDL structures are aligned the same way your C compiler will align them. This differs from one architecture to the next, so there may be different amounts of padding on different machines. This should be transparent, since both IDL and your C code are compiled and running on the same architecture. Note that IDL does NOT guarantee the way structures are stored and passed. This is the current behavior however.

3) A C union is just one location used to store 2 "things". IDL has no equivalent.

Note that CALL_EXTERNAL on VMS simply passes the arguments (by reference or descriptor) while on Unix the (argc, argv) mechanism is used.

Here is some C code which allows passing strings and works both on VMS and Unix with both IDL and PV-WAVE.

```
/*  ezcaIDL.c
*
```

```
* This file is an interface layer between IDL/PV-WAVE and EZCA.  It mainly
*   just converts parameter passing from the IDL/PV-WAVE mechanisms
*   (most parameters passed by reference) to the mechanism required by EZCA.
*   For a few the functions which are not supplied by EZCA this routine actually
*   does the work, using the ezcaPvToChid to fetch the channel descriptor.
*
* Author: Mark Rivers
* Date: June 28, 1995
*/

/* 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->length = strlen(s->address)
static char* str_array_addr[1000];
#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) {
#   define WAVE_HEADER5(ftype, fname, type1, arg1, type2, arg2, type3, arg3, type4, arg4,
type5, arg5)\
      ftype fname(type1 *arg1, type2 *arg2, type3 *arg3, type4 *arg4, type5 *arg5) {

#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];
#   define WAVE_HEADER5(ftype, fname, type1, arg1, type2, arg2, type3, arg3, type4, arg4,
type5, arg5)\
   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];\
    type5 *arg5 = (type5 *) argp[4];
#endif


WAVE_HEADER4(int, ezcaIDLGet, STRARG, pvname, char, type, int, nelem, void, buff)
  return(ezcaGet(STRADDR(pvname), *type, *nelem, buff));
}

WAVE_HEADER3(int, ezcaIDLGetControlLimits, STRARG, pvname, double, low, double, high)
  return(ezcaGetControlLimits(STRADDR(pvname), low, high));
}

WAVE_HEADER3(int, ezcaIDLGetGraphicLimits, STRARG, pvname, double, low, double, high)
  return(ezcaGetGraphicLimits(STRADDR(pvname), low, high));
}

WAVE_HEADER2(int, ezcaIDLGetPrecision, STRARG, pvname, short, precision)
  return(ezcaGetPrecision(STRADDR(pvname), precision));
}

WAVE_HEADER4(int, ezcaIDLGetStatus, STRARG, pvname, TS_STAMP, timestamp,
short, status, short, severity)
  return(ezcaGetStatus(STRADDR(pvname), timestamp, status, severity));
}

WAVE_HEADER2(int, ezcaIDLGetUnits, STRARG, pvname, char, units)
  return(ezcaGetUnits(STRADDR(pvname), units));
}

WAVE_HEADER1(int, ezcaIDLPerror, char, prefix)
  ezcaPerror(prefix);
  return EZCA_OK;
}
```

Subject: Re: Data passing & CALL_EXT
Posted by davidf on Fri, 21 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Bob Mallozzi writes:

> Is it possible to pass the following data structures from C code back
> to IDL via CALL_EXTERNAL:

>
> 1. An array of strings?

Yes. As byte arrays, usually.

> 2. A C structure, containing int, float, char

No. At least not officially. Check with Mark Rivers if you want
the unofficialy story.

> 3. A C union

Probably, as a byte array.

> I got my C code to pass back a single string, but could not do an
> array of strings. Really I would like to pass a structure back to
> IDL. Does anyone have any examples of this? The IDL examples treat
> only the simplest cases.

I'd ask Mark Rivers if he would help you if you bought him dinner
the next time you saw him.

> PS I am using OpenVMS :-(

Oh, well then ... prayer?

Cheers!

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Data passing & CALL_EXT
Posted by Achim Hein on Fri, 21 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

mallozzi@ssl.msfc.nasa.gov wrote:
>
> Is it possible to pass the following data structures from C code back
> to IDL via CALL_EXTERNAL:

>
> 1. An array of strings?

I can give you the code to pass 8Bit/16Bit data fields from C code back
to WAVE.

> 2. A C structure, containing int, float, char

...only back to WAVE.

> PS I am using OpenVMS :-(
Oh my God, there is another one using this incredible OpenVMS, so we are
two - and I have a question: Do you have any experience connecting VMS
machines to NT machines that form you can use your VMS licence manager
also within WindowsNT.

Regards

Achim
--

_____

Dipl.-Ing. A. Hein
PB2 / ZESS - Uni-GH-Siegen
Paul-Bonatz Str. 9-11
57068 Siegen
Phone: 0271/740-3362
Fax: 0271/740-2336
E-Mail: hein@nv.et-inf.uni-siegen.de

_____

Please have a look at our Web-Sites:

http://www.nv.et-inf.uni-siegen.de/pb2/www_pb2

_____

Subject: Re: Data passing & CALL_EXT
Posted by Phil Williams on Fri, 21 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

mallozzi@ssl.msfc.nasa.gov wrote:
>
> Hi,
>
> Is it possible to pass the following data structures from C code back
> to IDL via CALL_EXTERNAL:
>

> 1. An array of strings?
> 2. A C structure, containing int, float, char
> 3. A C union
>
> I got my C code to pass back a single string, but could not do an
> array of strings. Really I would like to pass a structure back to
> IDL. Does anyone have any examples of this? The IDL examples treat
> only the simplest cases.
>
> -bob
>
> PS I am using OpenVMS :-(

The returned variable from a call_external can only be a single value.
If you want to pass more than one then you must first create the var in
IDL and pass it to call_external. Then in your C function you place
your answer into this var.

Hope this helps,
Phil
--
/** ********/
Phil Williams, Ph.D.
Research Instructor
Children's Hospital Medical Center "One man gathers what
Imaging Research Center another man spills..."
3333 Burnet Ave. -The Grateful Dead
Cincinnati, OH 45229
email: williams@irc.chmcc.org
URL: http://scuttle.chmcc.org/~williams/
/** ********/