
Subject: Nested Structure DLM

Posted by [ibusoni](#) on Fri, 12 Jan 2007 15:10:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

To DLM's gurus.

I need to import in IDL a rather complex C++ structure:

object V000 containing a dictionary ("time","force") of n0 elements,

object V001 containing a dictionary ("time","force") of n1 elements,

....

object V00K containing a dictionary ("time","force") of nk elements,

The important point is that I don't know until run-time neither the number of objects K, nor the lengths of the dictionaries n0,..nk.

I set up a test program that pass back in its argument a structure FOO containing 5 structure V000 to V004.

Each one of the V00x contains 2 arrays "time" and "force" of different lengths, dblarr[3] for v000 up to dblarr[7] for v004.

Here's the code:

```
static IDL_MEMINT times_dims[] = { 1, 1 };
```

```
static IDL_MEMINT force_dims[] = { 1, 1 };
```

```
static IDL_STRUCT_TAG_DEF substruct_tags[] = {  
    {"TIME", times_dims, (void *) IDL_TYP_DOUBLE},  
    {"FORCE", force_dims, (void *) IDL_TYP_DOUBLE},  
    {0}  
};
```

```
IDL_VPTR IDLTestS(int Argc, IDL_VPTR Argv[])  
{
```

```
    IDL_VPTR  variabile = Argv[0];
```

```
    void *s;
```

```
    void *struct_s;
```

```
    IDL_MEMINT n_ele=1;
```

```
    // Dummy data
```

```
    int size=100000;
```

```
    double *s_data = (double*)malloc(size*sizeof(double));
```

```
    for (int i=0; i<size; i++){
```

```
        s_data[i]=i;
```

```
    }
```

```

// I need to create the IDL_STRUCT_TAG_DEF [] at run time
// because I don't know a priori the number of objects
int n_of_objects=5;
IDL_STRUCT_TAG_DEF *struct_tags =
    (IDL_STRUCT_TAG_DEF*)
malloc(sizeof(IDL_STRUCT_TAG_DEF) * (n_of_objects+1) );
IDL_STRUCT_TAG_DEF *tag;
for (int i=0; i<n_of_objects; i++){
    tag = &struct_tags[i];
    tag->name=(char*)malloc(5);
    snprintf(tag->name,5,"V%03d",i);
    tag->dims=(IDL_MEMINT*) malloc(2*sizeof(IDL_MEMINT));
    tag->dims[0]=1;
    tag->dims[1]=1;
    tag->type=NULL;
}
// terminating the array of IDL_STRUCT_TAG_DEF
tag = &struct_tags[n_of_objects];
tag->name=0;

// create substructs
for (int i=0; i<n_of_objects; i++){
    char nome[5];
    snprintf(nome,5,"V%03d",i);

    times_dims[1] = 3+i;
    force_dims[1] = 3+i;
    s = IDL_MakeStruct(nome, substruct_tags);
    struct_tags[i].type = s ;
}
// create main struct
struct_s = IDL_MakeStruct("FOO", struct_tags);

// see if IDL_STRUCT_TAG_DEF [] is correct
printf("printing struct_tags\n");
{
    int itag=0;
    IDL_STRUCT_TAG_DEF *tag = &struct_tags[itag];
    while ( ((char*)tag)[0] != 0) {
        printf("%s - (%ld %ld) - %p\n",tag->name, tag->dims[0],
tag->dims[1], tag->type);
        tag = &struct_tags[++itag];
    }
}

// attach data to the created structure
IDL_VPTR vv = IDL_ImportArray( 1, &n_ele, IDL_TYP_STRUCT, (UCHAR
*)s_data, idl_free_cb, struct_s);

```

```

IDL_VarCopy(vv, variabile);

return IDL_GettmpLong(1);
}

```

The main point is that I cannot define the IDL_STRUCT_TAG_DEF [] struct_tags at compile-time as usual, since I don't know the number of objects until run-time.

The output of the test is:

```

IDL> print, tests(a)
% Loaded DLM: TESTS.
printing struct_tags
V000 - (1 1) - 0x823dec4
V001 - (1 1) - 0x823dfbc
V002 - (1 1) - 0x823e0b4
V003 - (1 1) - 0x823e1ac
V004 - (1 1) - 0x823e2a4
      1
IDL> help ,a , /str
** Structure FOO, 7 tags, length=400, data length=400:
  TIME      DOUBLE  Array[3]
  FORCE      DOUBLE  Array[3]
  V001      STRUCT  -> V001 Array[1]
  V002      STRUCT  -> V002 Array[1]
  TIME      DOUBLE  Array[6]
  FORCE      DOUBLE  Array[6]
  V004      STRUCT  -> V004 Array[1]
IDL> help ,a.v001 , /str
** Structure V001, 2 tags, length=64, data length=64:
  TIME      DOUBLE  Array[4]
  FORCE      DOUBLE  Array[4]
IDL> print, a
{   0.0000000   1.0000000   2.0000000
   3.0000000   4.0000000   5.0000000
{
   6.0000000   7.0000000   8.0000000   9.0000000
  10.000000   11.000000   12.000000   13.000000
}{
  14.000000   15.000000   16.000000   17.000000
18.000000
  19.000000   20.000000   21.000000   22.000000
23.000000
}
  24.000000   25.000000   26.000000   27.000000

```

```
28.000000    29.000000
    30.000000    31.000000    32.000000    33.000000
34.000000    35.000000
{
    36.000000    37.000000    38.000000    39.000000
40.000000    41.000000    42.000000
    43.000000    44.000000    45.000000    46.000000
47.000000    48.000000    49.000000
}}
IDL>
```

Any idea of why some of the substructures (v000 and v003) have not been properly setup????

How can FOO have 2 fields TIME and 2 fields FORCE???

This behaviour is reproducible, but the "missing" substructures can change (i.e. instead of V000 and V003 there can be other "missing" substructures).

It seems IDL_MakeStruct got confused by my messy code :(
Is there a different way of doing this ???

Thanks
Lorenzo
