Subject: Re: DLM Woes
Posted by Karl[1] on Thu, 27 Jan 2011 22:50:24 GMT
View Forum Message <> Reply to Message

On Jan 27, 12:30 pm, Mort Canty <m.ca...@fz-juelich.de> wrote:
> Hi Ronn,
>
> Here's the whole thing (not very long):
>
> void IDL_CDECL cuda_svd(int argc, IDL_VPTR argv[])
> {
> // output array pointers
> float * w, * u, * vt;
> // get the input matrix
> float * a = (float *) argv[0]->value.arr->data;
> // get its dimensions
> int ndim_a = (int) argv[0]->value.arr->n_dim;
> IDL_LONG64 * dim_a = argv[0]->value.arr->dim;
> int m = (int) dim_a[0];
> int n = (int) dim_a[1];
>
> // IDL output arrays
> IDL_VPTR ivWptr, ivUptr, ivVTptr;
> int ndim_w = 1;
> IDL_LONG64 dim_w[] = {min(m,n)};
> IDL_LONG64 dim_u[] = {m,min(m,n)};
> IDL_LONG64 dim_v[] = {min(m,n),n};
> w = (float *) IDL_MakeTempArray((int) IDL_TYP_FLOAT, ndim_w, dim_w,
> IDL_ARR_INI_ZERO, &ivWptr);
> u = (float *) IDL_MakeTempArray((int) IDL_TYP_FLOAT, ndim_a,
> dim_u, IDL_ARR_INI_ZERO, &ivUptr);
> vt= (float *) IDL_MakeTempArray((int) IDL_TYP_FLOAT, ndim_a, dim_v,
> IDL_ARR_INI_ZERO, &ivVTptr);
>
> // CULA general matrix single precision SVD with host pointers
> culaStatus s = culaInitialize();
> if(s == culaNoError)
> {
> s = culaSgesvd('S','S',m,n,a,m,w,u,m,vt,min(m,n));
> culaShutdown();
> }
> // return results to IDL (all zeroes if CULA failed to initialize)
> IDL_VarCopy(ivWptr,argv[1]);
> IDL_VarCopy(ivVTptr,argv[2]);
> IDL_VarCopy(ivUptr,argv[3]);
> }
>
> Here is what the compiler puts out:

>
> 1>------ Build started: Project: cuda_svd, Configuration: Debug x64 ------
> 1> cuda_svd.c
> 1>D:\Idl\projects\development\svd\cuda_svd.c(50): error C2275:
> 'culaStatus' : illegal use of this type as an expression
> 1> c:\program files\cula\include\culastatus.h(63) : see
> declaration of 'culaStatus'
> 1>D:\Idl\projects\development\svd\cuda_svd.c(50): error C2146: syntax
> error : missing ';' before identifier 's'
> 1>D:\Idl\projects\development\svd\cuda_svd.c(50): error C2065: 's' :
> undeclared identifier
> 1>D:\Idl\projects\development\svd\cuda_svd.c(51): error C2065: 's' :
> undeclared identifier
> 1>D:\Idl\projects\development\svd\cuda_svd.c(53): error C2065: 's' :
> undeclared identifier
> ========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
>
> I discovered if I move the statement
>
> culaStatus s = culaInitialize();
>
> upwards, preceding the comment line
>
> // IDL output arrays
>
> the damn thing builds (and runs correctly under IDL)! Way over my head,
> I'm afraid.
>
> Thanks,
>
> Mort
>
> Am 27.01.2011 19:49, schrieb ronn kling:
>
>> Hi Mort,
>
>> Can you show me how you have declared the variables in the call?
>
>> Ronn Kling
>
>> "Mort Canty" <m.ca...@fz-juelich.de> wrote in message
>> news:ihrlro$5n8j$1@zam602.zam.kfa-juelich.de...
>>> Apologies in advance: This is a C question, but I know a lot of you
>>> write your own DLMs for IDL and I'm completely stumped. I have a DLM
>>> with this statement, patterned after Ronn Kling's book, near the
>>> middle of the code:
>
>>> w = (float *) IDL_MakeTempArray((int) IDL_TYP_FLOAT, ndim_w, dim_w,

>>> IDL_ARR_INI_ZERO, &ivWptr);
>
>>> The DLM compiles and runs perfectly on the win32 platform. When I try
>>> to compile the exact same code to the x64 platform, all of the
>>> statements which follow the above one are riddled with crazy syntax
>>> errors. All those preceding it are accepted. I'm using Visual C++ 2010
>>> Express with the Windows SDK Version 7.1.
>
>>> Any help for a complete C novice?
>
>>> Thanks
>
>>> Mort
>
>

Sort of looks like you are declaring a variable, s, in the middle of a
code block. To enable this to compile, you have to compile the C code
in C++ mode, or set a compiler option to allow this usage. Or, you
can put

culaStatus s;

up above the comment line:

// IDL output arrays

and then just say:

s = culaInitialize();

Somehow you were compiling the code with different options for 32-bit
and 64-bit.

Either fix the code to make it legal C (I know some newer C standards
allow this) or set your project file up to use the same compiler
options for 32-bit and 64-bit. My guess is that the x64 Debug config
does not have the "compile as C++" option set.
