Subject: Re: IDL/DLM segmentation fault on reset
Posted by Haje Korth on Tue, 31 Aug 2010 20:08:56 GMT
View Forum Message <> Reply to Message

Just FYI: I confirm that replacing putenv() with setenv() avoids the segmentation fault on IDL session reset. I do not think this routine exists in Visual Studio. However, putenv did not create any issues on windows.

Haje

On Aug 31, 11:24 am, Haje Korth <hajeko...@gmail.com> wrote:
> Hi Karl,
> thanks so much for your reply. You are exactly right, the variable
> input to putenv is a static char*, and your description of what is
> going on makes sense. I will give setenv a shot later today and report
> back.
>
> Cheers,
> haje
>
> On Aug 31, 10:54 am, Karl <karl.w.schu...@gmail.com> wrote:
>
>> On Aug 30, 6:34 pm, Haje Korth <hajeko...@gmail.com> wrote:
>
>>> Hi,
>>> Have an interesting problem for DLM programmers out there. I have a C
>>> code for Linux that sets an environment variable using putenv() within
>>> a DLM. The code executes as designed but when I issue
>>> a .full_session_reset I get a segmentation fault and IDL exits
>>> ungracefully. I run the same code (with slightly different calling
>>> convention) under Windows without problems. Does anyone know why? Do I
>>> need some sort of exit handler for the DLM (similar to IDL_Load for
>>> startup)?
>
>>> Haje
>
>> The putenv() function just puts a pointer to the string in the
>> environment. It does not make a copy of the string.
>
>> The full session reset unloads the DLM. Your string argument to
>> putenv() probably resides in a static var in the DLM. The environment
>> is just a list of pointers to strings. When the DLM got unloaded, the
>> environment is left with an invalid pointer to a string. I can't say
>> exactly what was going on when the segmentation fault occurred, but it
>> is easy to imagine that some system routine running during the session
>> reset was probably walking the environment strings and hit the bogus
>> pointer.

>
>> You might try using setenv() instead. The setenv() function allocates
>> heap memory for environment strings, and thus avoids the string memory
>> from getting unallocated by a unload of a module or an automatic
>> variable going out of scope.
>
>> It worked on Windows because Windows probably stores the environment
>> strings differently.
>
>> Karl
>
>
