Subject: Re: A simple DLM question
Posted by Jim Pendleton on Mon, 27 Aug 2012 20:48:26 GMT
View Forum Message <> Reply to Message

On Monday, August 27, 2012 11:13:57 AM UTC-6, Xin Tao wrote:
> Hi,
>
>
>
> I'm having trouble figuring out the problem of the following DLM code:
>
>
>
> /* The c routine */
>
>
>
> void simple(int argc, IDL_VPTR argv[])
>
> {
>
> IDL_VPTR v;
>
>
>
> v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>
>
> IDL_DELTMP(v);
>
> }
>
>
>
> This routine just takes its input and convert it to double. After converting it to a DLM, however, I seem to see strange results.
>
>
>
> IDL> simple, 1.0d
>
> % Loaded DLM: TESTMODULE.
>
> IDL> simple, -1.0d
>
> Bus error
>

>
>
> That is: if I give it 1.0d as input, then the code is fine. However, if I use -1.0d, then there is a BUS error, presumably from IDL_DELTMP(v). I really don't understand why this is the case. Isn't IDL_DELTMP supposed to decide first whether v is a temporary variable or not? If I remove IDL_DELTMP, of course, I'll frequently get the annoying warning message "% Temporary variables are still checked out - cleaning up...".
>
>
>
> Please give me some help. Thanks.

Try this:

if (v != argv[0]) IDL_DELTMP(v);

That is, no conversion was necessary.

The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants.

Subject: Re: A simple DLM question
Posted by Craig Markwardt on Mon, 27 Aug 2012 20:51:10 GMT
View Forum Message <> Reply to Message

On Monday, August 27, 2012 1:13:57 PM UTC-4, Xin Tao wrote:
...
> Please give me some help. Thanks.

Your first problem is putting "Simple" and "DLM question" in the same sentence. :-)

Subject: Re: A simple DLM question
Posted by Xin Tao on Mon, 27 Aug 2012 22:04:43 GMT
View Forum Message <> Reply to Message

Thanks Jimmy. That indeed solved my problem. It was so confusing to me, because I found from the External Development Guide that IDL_DELTMP should check it first. :)

On Monday, August 27, 2012 3:48:27 PM UTC-5, jimmylee...@gmail.com wrote:
> On Monday, August 27, 2012 11:13:57 AM UTC-6, Xin Tao wrote:
>
>> Hi,
>
>>
>

>>
>
>>
>
>> I'm having trouble figuring out the problem of the following DLM code:
>
>>
>
>>
>
>>
>
>> /* The c routine */
>
>>
>
>>
>
>>
>
>> void simple(int argc, IDL_VPTR argv[])
>
>>
>
>> {
>
>>
>
>> IDL_VPTR v;
>
>>
>
>>
>
>>
>
>>
>
>> v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>>
>
>>
>
>>
>
>> IDL_DELTMP(v);
>
>>
>

>> }
>
>>
>
>>
>
>>
>
>> This routine just takes its input and convert it to double. After converting it to a DLM, however, I seem to see strange results.
>
>>
>
>>
>
>>
>
>> IDL> simple, 1.0d
>
>>
>
>> % Loaded DLM: TESTMODULE.
>
>>
>
>> IDL> simple, -1.0d
>
>>
>
>> Bus error
>
>>
>
>>
>
>>
>
>> That is: if I give it 1.0d as input, then the code is fine. However, if I use -1.0d, then there is a BUS error, presumably from IDL_DELTMP(v). I really don't understand why this is the case. Isn't IDL_DELTMP supposed to decide first whether v is a temporary variable or not? If I remove IDL_DELTMP, of course, I'll frequently get the annoying warning message "% Temporary variables are still checked out - cleaning up...".
>
>>
>
>>
>
>>

>
>> Please give me some help. Thanks.
>
>
>
> Try this:
>
>
>
> if (v != argv[0]) IDL_DELTMP(v);
>
>
>
> That is, no conversion was necessary.
>
>
>
> The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants.

Subject: Re: A simple DLM question
Posted by Xin Tao on Mon, 27 Aug 2012 22:08:27 GMT
View Forum Message <> Reply to Message

On Monday, August 27, 2012 3:51:10 PM UTC-5, Craig Markwardt wrote:
> On Monday, August 27, 2012 1:13:57 PM UTC-4, Xin Tao wrote:
>
> ...
>
>> Please give me some help. Thanks.
>
>
>
> Your first problem is putting "Simple" and "DLM question" in the same sentence. :-)

I've been learning DLM for a whole day, and I thought that would be a *simple* question for some experts. :)

Subject: Re: A simple DLM question
Posted by Jim Pendleton on Tue, 28 Aug 2012 16:06:44 GMT
View Forum Message <> Reply to Message

On Monday, August 27, 2012 4:04:43 PM UTC-6, Xin Tao wrote:
> Thanks Jimmy. That indeed solved my problem. It was so confusing to me, because I found from the External Development Guide that IDL_DELTMP should check it first. :)

>
>
>
> On Monday, August 27, 2012 3:48:27 PM UTC-5, jimmylee...@gmail.com wrote:
>
>> On Monday, August 27, 2012 11:13:57 AM UTC-6, Xin Tao wrote:
>
>>
>
>>> Hi,
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I'm having trouble figuring out the problem of the following DLM code:
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> /* The c routine */
>
>>
>
>>>
>
>>

```
>
>>>
>
>>
>
>>>
>
>>
>
>>>  void simple(int argc, IDL_VPTR argv[])
>
>>
>
>>>
>
>>
>
>>>  {
>
>>
>
>>>
>
>>
>
>>>    IDL_VPTR v;
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>    v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>>
>
>>>
>
>>
```

>
>>>
>
>>
>
>>>
>
>>
>
>>>   IDL_DELTMP(v);
>
>>
>
>>>
>
>>
>
>>> }
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> This routine just takes its input and convert it to double. After converting it to a DLM, however, I seem to see strange results.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>

>>
>
>>> IDL> simple, 1.0d
>
>>
>
>>>
>
>>
>
>>> % Loaded DLM: TESTMODULE.
>
>>
>
>>>
>
>>
>
>>> IDL> simple, -1.0d
>
>>
>
>>>
>
>>
>
>>> Bus error
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> That is: if I give it 1.0d as input, then the code is fine. However, if I use -1.0d, then there is a BUS error, presumably from IDL_DELTMP(v). I really don't understand why this is the case. Isn't IDL_DELTMP supposed to decide first whether v is a temporary variable or not? If I remove IDL_DELTMP, of course, I'll frequently get the annoying warning message "% Temporary variables are still checked out - cleaning up...".
>

>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Please give me some help. Thanks.
>
>>
>
>>
>
>>
>
>> Try this:
>
>>
>
>>
>
>>
>
>> if (v != argv[0]) IDL_DELTMP(v);
>
>>
>
>>
>
>>
>
>> That is, no conversion was necessary.
>
>>
>
>>
>
>>
>
>> The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants.

The docs are correct, but are confusing if you're not aware of the difference between IDL's temporary variables, constants, and named variables. It's not stated explicitly in this section that a constant like 1.0D is a different sort of data type internally than an expression or named variable, though that topic is discussed earlier in the docs.

IDL_DELTMP doesn't check if the IDL_VARIABLE has the contant flag set (IDL_V_CONST), only the temporary flag (IDL_V_TEMP).

As a matter of habit, I always check the equality of the argv[] used as input against the output from any type conversion routine call before calling IDL_DELTMP. You can't predict when a user has entered an explicit constant value, rather than a variable name or expression.

Subject: Re: A simple DLM question
Posted by Xin Tao on Tue, 28 Aug 2012 21:16:53 GMT
View Forum Message <> Reply to Message

I did check the flags of IDL_V_CONST and IDL_V_TEMP. Both failed, and then I posted the question. For example, I'm not sure how to explain this results.

// Code here.

```
 void simple(int argc, IDL_VPTR argv[])
{
  IDL_VPTR v;

  v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);

  printf("const = %d\n", v->flags & IDL_V_CONST);
  printf("temp = %d\n", v->flags & IDL_V_TEMP);

  if (v != argv[0]) IDL_DELTMP(v);
}
```

Now results:

```
IDL> simple, 3.0d
const = 1
temp = 0
IDL> simple, -3.0d
const = 0
temp = 2
```

This sounds really strange to me. But if there is a good explanation of this, please let me know. Thanks.

Xin Tao

On Tuesday, August 28, 2012 11:06:44 AM UTC-5, jimmylee...@gmail.com wrote:
> On Monday, August 27, 2012 4:04:43 PM UTC-6, Xin Tao wrote:
>
>> Thanks Jimmy. That indeed solved my problem. It was so confusing to me, because I found from the External Development Guide that IDL_DELTMP should check it first. :)
>
>>
>
>>
>
>>
>
>> On Monday, August 27, 2012 3:48:27 PM UTC-5, jimmylee...@gmail.com wrote:
>
>>
>
>>> On Monday, August 27, 2012 11:13:57 AM UTC-6, Xin Tao wrote:
>
>>
>
>>>
>
>>
>
>>>> Hi,
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>

>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> I'm having trouble figuring out the problem of the following DLM code:
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> /* The c routine */
>
>>
>

```
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  void simple(int argc, IDL_VPTR argv[])
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> {
>
>>
>
```

```
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>   IDL_VPTR v;
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>   v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>>
>
```

```
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>   IDL_DELTMP(v);
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> }
>
>>
>
```

>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> This routine just takes its input and convert it to double. After converting it to a DLM, however, I seem to see strange results.
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>

>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> IDL> simple, 1.0d
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> % Loaded DLM: TESTMODULE.
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> IDL> simple, -1.0d
>
>>

>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> Bus error
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> That is: if I give it 1.0d as input, then the code is fine. However, if I use -1.0d, then there is a BUS error, presumably from IDL_DELTMP(v). I really don't understand why this is the case. Isn't IDL_DELTMP supposed to decide first whether v is a temporary variable or not? If I remove

IDL_DELTMP, of course, I'll frequently get the annoying warning message "% Temporary variables are still checked out - cleaning up...".
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>> Please give me some help. Thanks.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>

>
>>> Try this:
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> if (v != argv[0]) IDL_DELTMP(v);
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> That is, no conversion was necessary.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>

>
>>> The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants.
>
>
>
> The docs are correct, but are confusing if you're not aware of the difference between IDL's temporary variables, constants, and named variables. It's not stated explicitly in this section that a constant like 1.0D is a different sort of data type internally than an expression or named variable, though that topic is discussed earlier in the docs.
>
>
>
> IDL_DELTMP doesn't check if the IDL_VARIABLE has the contant flag set (IDL_V_CONST), only the temporary flag (IDL_V_TEMP).
>
>
>
> As a matter of habit, I always check the equality of the argv[] used as input against the output from any type conversion routine call before calling IDL_DELTMP. You can't predict when a user has entered an explicit constant value, rather than a variable name or expression.

Subject: Re: A simple DLM question
Posted by Craig Markwardt on Wed, 29 Aug 2012 01:10:17 GMT
View Forum Message <> Reply to Message

On Tuesday, August 28, 2012 5:16:53 PM UTC-4, Xin Tao wrote:
> I did check the flags of IDL_V_CONST and IDL_V_TEMP. Both failed, and then I posted the question. For example, I'm not sure how to explain this results.
>
>
>
> // Code here.
>
>
>
> void simple(int argc, IDL_VPTR argv[])
>
> {
>
> IDL_VPTR v;
>
>
>
> v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>

>
> printf("const = %d\n", v->flags & IDL_V_CONST);
>
> printf("temp = %d\n", v->flags & IDL_V_TEMP);
>
>
>
> if (v != argv[0]) IDL_DELTMP(v);
>
> }
>
>
>
>
>
>
> Now results:
>
>
>
> IDL> simple, 3.0d
>
> const = 1
>
> temp = 0
>
> IDL> simple, -3.0d
>
> const = 0
>
> temp = 2

It is a little odd, but it may be the difference between 3.0d, a constant, and -(3.0d), which is an expression. But I'd think that IDL should be smart enough to parse -3.0d as a constant.

Craig

Subject: Re: A simple DLM question
Posted by Xin Tao on Wed, 29 Aug 2012 05:03:27 GMT
View Forum Message <> Reply to Message

Maybe it's a bug! Anyway, Jimmy's trick did it, but I'm still confused by IDL's EDG.

On Tuesday, August 28, 2012 8:10:17 PM UTC-5, Craig Markwardt wrote:
> On Tuesday, August 28, 2012 5:16:53 PM UTC-4, Xin Tao wrote:
>
>> I did check the flags of IDL_V_CONST and IDL_V_TEMP. Both failed, and then I posted the question. For example, I'm not sure how to explain this results.

```
>
>>
>
>>
>
>>
>
>>  // Code here.
>
>>
>
>>
>
>>
>
>>   void simple(int argc, IDL_VPTR argv[])
>
>>
>
>> {
>
>>
>
>>    IDL_VPTR v;
>
>>
>
>>
>
>>
>
>>    v = IDL_BasicTypeConversion(1, &argv[0], IDL_TYP_DOUBLE);
>
>>
>
>>
>
>>
>
>>    printf("const = %d\n", v->flags & IDL_V_CONST);
>
>>
>
>>    printf("temp = %d\n", v->flags & IDL_V_TEMP);
>
>>
>
>>
```

>
>>
>
>>   if (v != argv[0]) IDL_DELTMP(v);
>
>>
>
>> }
>
>>
>
>>
>
>>
>
>>
>
>>
>
>> Now results:
>
>>
>
>>
>
>>
>
>> IDL> simple, 3.0d
>
>>
>
>> const = 1
>
>>
>
>> temp = 0
>
>>
>
>> IDL> simple, -3.0d
>
>>
>
>> const = 0
>
>>
>
>> temp = 2

>
>
>
> It is a little odd, but it may be the difference between 3.0d, a constant, and -(3.0d), which is an expression. But I'd think that IDL should be smart enough to parse -3.0d as a constant.
>
>
>
> Craig

Subject: Re: A simple DLM question
Posted by chris_torrence@NOSPAM on Wed, 29 Aug 2012 15:55:28 GMT
View Forum Message <> Reply to Message

Hi all,

Just to clarify, this isn't a bug. It just happens that since the minus sign is an operator, then a number such as -1 gets turned into an expression, which is stored in a temporary variable. It is treated the same way as say "-a" where "a" is a variable.

Jimmy said "The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants."

That isn't quite correct. In this particular case, you *must* check (v != argv[0]), regardless of whether it is a constant or temp variable. That is because if the type conversion was not needed, then "v" will be equal to argv[0]. In this case, your code does not "own" argv[0] since it didn't allocate it. Freeing "v" is bad in that case, because IDL will attempt to free it later, and a double free will occur.

Probably the best way to write the line of code is:

if (v != argv[0]) IDL_Deltmp(v);

Here, you don't need to use the macro, because you *know* that if "v" is not equal, then it must be a temporary that is owned by you, and you can free it without doing any further checks.

Hope this helps.

-Chris
ExelisVIS

Subject: Re: A simple DLM question
Posted by Xin Tao on Wed, 29 Aug 2012 16:23:51 GMT
View Forum Message <> Reply to Message

That's very clear explanation, thanks. But I would still say that it's not very intuitive to me that 3 is a constant but -3 is not.

On Wednesday, August 29, 2012 10:55:28 AM UTC-5, Chris Torrence wrote:
> Hi all,
>
>
>
> Just to clarify, this isn't a bug. It just happens that since the minus sign is an operator, then a number such as -1 gets turned into an expression, which is stored in a temporary variable. It is treated the same way as say "-a" where "a" is a variable.
>
>
>
> Jimmy said "The macro (in idl_export.h, if you're interested) doesn't do extensive checking, and you should only free variables that are temps, not expressions or constants."
>
>
>
> That isn't quite correct. In this particular case, you *must* check (v != argv[0]), regardless of whether it is a constant or temp variable. That is because if the type conversion was not needed, then "v" will be equal to argv[0]. In this case, your code does not "own" argv[0] since it didn't allocate it. Freeing "v" is bad in that case, because IDL will attempt to free it later, and a double free will occur.
>
>
>
> Probably the best way to write the line of code is:
>
>
>
> if (v != argv[0]) IDL_Deltmp(v);
>
>
>
> Here, you don't need to use the macro, because you *know* that if "v" is not equal, then it must be a temporary that is owned by you, and you can free it without doing any further checks.
>
>
>
> Hope this helps.
>
>
>
> -Chris
>
> ExelisVIS

Subject: Re: A simple DLM question
Posted by Lajos Foldy on Wed, 29 Aug 2012 17:31:38 GMT
View Forum Message <> Reply to Message

On Wednesday, August 29, 2012 6:23:51 PM UTC+2, Xin Tao wrote:

> That's very clear explanation, thanks. But I would still say that it's not very intuitive to me that 3 is a constant but -3 is not.

A '-' can be both unary (-3) and binary (a-3). Including the '-' in the constant would be erroneous in the binary case and you would get a syntax error.

There is a similar feature (or bug?) in parsing the exponent of floating point literals. IDL accepts 1e- as a valid float number:

```
IDL> help, 1e-
<Expression>    FLOAT     =       1.00000
```

But this results in a syntax error for expressions like 1e-a:

```
IDL> a=0 & help, 1e-a

a=0 & help, 1e-a
              ^
% Syntax error.
```

(I think 1e- should be invalid and 1e-a should be valid.)

regards,
Lajos

Subject: Re: A simple DLM question
Posted by chris_torrence@NOSPAM on Wed, 29 Aug 2012 17:50:33 GMT
View Forum Message <> Reply to Message

On Wednesday, August 29, 2012 11:31:39 AM UTC-6, fawltyl...@gmail.com wrote:
>
> A '-' can be both unary (-3) and binary (a-3). Including the '-' in the constant would be erroneous in the binary case and you would get a syntax error.
>
> There is a similar feature (or bug?) in parsing the exponent of floating point literals. IDL accepts 1e- as a valid float number:
>
> IDL> help, 1e-
> <Expression>   FLOAT   =   1.00000
>

> But this results in a syntax error for expressions like 1e-a:
>
> IDL> a=0 & help, 1e-a
>
> a=0 & help, 1e-a
>   ^
> % Syntax error.
>
> (I think 1e- should be invalid and 1e-a should be valid.)
>
> regards,
> Lajos

This is the one that always drives me nuts:
IDL> a = 3
IDL> help,1d+a
% Syntax error.
IDL> help,1d + a
<Expression>   DOUBLE   =   4.0000000

What kind of crazy language requires spaces to work properly?!
[Three cheers for whoever answers this question first! And IDL doesn't count...]

-Chris

Subject: Re: A simple DLM question
Posted by Michael Galloy on Wed, 29 Aug 2012 18:16:41 GMT
View Forum Message <> Reply to Message

On 8/29/12 11:50 AM, Chris Torrence wrote:
> On Wednesday, August 29, 2012 11:31:39 AM UTC-6, fawltyl...@gmail.com wrote:
>>
>> A '-' can be both unary (-3) and binary (a-3). Including the '-' in the constant would be erroneous in the binary case and you would get a syntax error.
>>
>> There is a similar feature (or bug?) in parsing the exponent of floating point literals. IDL accepts 1e- as a valid float number:
>>
>> IDL> help, 1e-
>> <Expression>   FLOAT   =   1.00000
>>
>> But this results in a syntax error for expressions like 1e-a:
>>
>> IDL> a=0 & help, 1e-a
>>
>> a=0 & help, 1e-a
>>   ^

>> % Syntax error.
>>
>> (I think 1e- should be invalid and 1e-a should be valid.)
>>
>> regards,
>> Lajos
>
> This is the one that always drives me nuts:
> IDL> a = 3
> IDL> help,1d+a
> % Syntax error.
> IDL> help,1d + a
> <Expression> DOUBLE = 4.0000000
>
> What kind of crazy language requires spaces to work properly?!
> [Three cheers for whoever answers this question first! And IDL doesn't count...]
>
> -Chris

Python! (Although you can use tabs as long as you are consistent.)

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation
