Subject: yet another idl memory question
Posted by jw2519 on Thu, 04 Jun 2009 15:53:05 GMT
View Forum Message <> Reply to Message

How do I force that IDL set aside a chunk of contiguous memory to be refilled by a variable of equal size at each iteration of a procedure?

More specifically, I have a code which consists of three procedures. One of these serves as the wrapper, one uses the IDL OpenDAP utility to download the relevant data, and the third processes the data into the grid and format that I want. The code will iterate for as long as the wrapper indicates that processed data files need to be created. I am running this on a 64-bit linux machine (IDL 7.0).

Each iteration of the code performs four OpenDAP calls; each call downloads a large (~750MB) structure. After several iterations, the code exits with an insufficient memory to allocate array error (or sometimes "error reading value array: read is for xxx bytes, but there are only yyy in the buffer"; always this error occurs within an OpenDAP call). Using help,/memory and several of the tricks and tools posted elsewhere, I have been able to substantially reduce the max memory usage, but this only nets me another one and a half iterations or so. So my guess is that the problem is that there is no longer a sufficiently large block of contiguous memory into which to download the structure. Each of the four structures is of equal size at each iteration. How can I set aside a block of contiguous memory so that IDL will reuse the same chunk for each structure every time i call the procedure? I've tried using a solitary common block (i.e. just within the procedure in question), and I've attempted to use pointers. The program runs in both cases, but still exits with the same error. Of course, I am not at all certain that I used either of these correctly...

Any suggestions are greatly appreciated.

Subject: Re: yet another idl memory question
Posted by David Fanning on Mon, 08 Jun 2009 19:05:06 GMT
View Forum Message <> Reply to Message

Paolo writes:

>> Of course, data=3D0B doesn't free *all* the memory,
>> and doing this many times leads, I suspect, to the memory
>> fragmentation that is the heart of the problem. I suggest
>> you use UNDEFINE. That really does release *all* the memory
>> associated with a variable.
>

> Really? Certainly it's not a substitute for ptr_free, is it?
> At least not in my system:

Well, I think you are confusing "variable", which is what
I claim, with "pointer to a variable", which I admit UNDEFINE
doesn't free. (I think it was written *before* pointers, to
tell you the truth!)

But in any case, easily fixed. Just test to see if the
variable is a pointer or object, destroy it if so, and
carry on undefining the variable.

Maybe I'll get around to it later today. :-)

Cheers,

David
--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by pgrigis on Mon, 08 Jun 2009 19:56:58 GMT
View Forum Message <> Reply to Message

On Jun 8, 3:05 pm, David Fanning <n...@dfanning.com> wrote:
> Paolo writes:
>>> Of course, data=3D0B doesn't free *all* the memory,
>>> and doing this many times leads, I suspect, to the memory
>>> fragmentation that is the heart of the problem. I suggest
>>> you use UNDEFINE. That really does release *all* the memory
>>> associated with a variable.
>
>> Really? Certainly it's not a substitute for ptr_free, is it?
>> At least not in my system:
>
> Well, I think you are confusing "variable", which is what
> I claim, with "pointer to a variable", which I admit UNDEFINE
> doesn't free. (I think it was written *before* pointers, to
> tell you the truth!)

Well, for me pointers are just another kind of variables :)

In fact, I don't think IDL has anything like classic
"pointers to variables": pointers are just references to
data in memory (similar to regular variabes), because a

command such as ptr_new(A) just duplicate the contents of A
to a new memory location, so there is no such a thing as a
true pointer to variable A, right?

Ciao,
Paolo

>
> But in any case, easily fixed. Just test to see if the
> variable is a pointer or object, destroy it if so, and
> carry on undefining the variable.
>
> Maybe I'll get around to it later today. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Coyote's Guide to IDL Programming (www.dfanning.com)
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by pgrigis on Mon, 08 Jun 2009 19:57:19 GMT
View Forum Message <> Reply to Message

On Jun 8, 3:05 pm, David Fanning <n...@dfanning.com> wrote:
> Paolo writes:
>>> Of course, data=3D0B doesn't free *all* the memory,
>>> and doing this many times leads, I suspect, to the memory
>>> fragmentation that is the heart of the problem. I suggest
>>> you use UNDEFINE. That really does release *all* the memory
>>> associated with a variable.
>
>> Really? Certainly it's not a substitute for ptr_free, is it?
>> At least not in my system:
>
> Well, I think you are confusing "variable", which is what
> I claim, with "pointer to a variable", which I admit UNDEFINE
> doesn't free. (I think it was written *before* pointers, to
> tell you the truth!)

Well, for me pointers are just another kind of variables :)

In fact, I don't think IDL has anything like classic
"pointers to variables": pointers are just references to
data in memory (similar to regular variabes), because a

command such as ptr_new(A) just duplicate the contents of A
to a new memory location, so there is no such a thing as a
true pointer to variable A, right?

Ciao,
Paolo

>
> But in any case, easily fixed. Just test to see if the
> variable is a pointer or object, destroy it if so, and
> carry on undefining the variable.
>
> Maybe I'll get around to it later today. :-)
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.
> Coyote's Guide to IDL Programming (www.dfanning.com)
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by David Fanning on Mon, 08 Jun 2009 20:09:40 GMT
View Forum Message <> Reply to Message

Paolo writes:

> In fact, I don't think IDL has anything like classic
> "pointers to variables": pointers are just references to
> data in memory (similar to regular variabes), because a
> command such as ptr_new(A) just duplicate the contents of A
> to a new memory location, so there is no such a thing as a
> true pointer to variable A, right?

If you like, you can think of a pointer as a type of
IDL variable that doesn't currently work with UNDEFINE. :-)

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by David Fanning on Wed, 10 Jun 2009 21:15:25 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> If you like, you can think of a pointer as a type of
> IDL variable that doesn't currently work with UNDEFINE. :-)

UNDEFINE has now been updated to recursively search
pointers, objects, and structure variables for other
variables it can destroy and devour before it gets
around to undefining the variable in question.

Even Paolo should be satisfied now. :-)

http://www.dfanning.com/programs/undefine.pro

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by pgrigis on Wed, 10 Jun 2009 22:01:22 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> David Fanning writes:
>
>> If you like, you can think of a pointer as a type of
>> IDL variable that doesn't currently work with UNDEFINE. :-)
>
> UNDEFINE has now been updated to recursively search
> pointers, objects, and structure variables for other
> variables it can destroy and devour before it gets
> around to undefining the variable in question.
>
> Even Paolo should be satisfied now. :-)
>
> http://www.dfanning.com/programs/undefine.pro

Cool! Way better then the kitchen sink to get rid of these
old smelly variables... :)

Ciao,
Paolo

>
> Cheers,
>
> David
>
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: yet another idl memory question
Posted by JDS on Mon, 15 Jun 2009 18:57:07 GMT
View Forum Message <> Reply to Message

Not to backseat code... but you can use HEAP_FREE for the heavy
lifting, and SCOPE_VARFETCH to save some bytes of disk space:

```
pro undef,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,v13,v14,v15
  for i=1,n_params() do begin
    name='V'+strtrim(i,2)
    heap_free,scope_varfetch(name)
    void=temporary(scope_varfetch(name))
  endfor
end
```

IDL doesn't have the concept of the ARGV pointer, or argument list
which can be iterated through, but sometimes SCOPE_VARFETCH and/or
SCOPE_VARNAME can do the trick.

Subject: Re: yet another idl memory question
Posted by ben.bighair on Mon, 15 Jun 2009 19:07:46 GMT
View Forum Message <> Reply to Message

On Jun 8, 3:57 pm, Paolo <pgri...@gmail.com> wrote:
> On Jun 8, 3:05 pm, David Fanning <n...@dfanning.com> wrote:
>

>> Paolo writes:
>>>> Of course, data=3D0B doesn't free *all* the memory,
>>>> and doing this many times leads, I suspect, to the memory
>>>> fragmentation that is the heart of the problem. I suggest
>>>> you use UNDEFINE. That really does release *all* the memory
>>>> associated with a variable.
>
>>> Really? Certainly it's not a substitute for ptr_free, is it?
>>> At least not in my system:
>
>> Well, I think you are confusing "variable", which is what
>> I claim, with "pointer to a variable", which I admit UNDEFINE
>> doesn't free. (I think it was written *before* pointers, to
>> tell you the truth!)
>
> Well, for me pointers are just another kind of variables :)
>
> In fact, I don't think IDL has anything like classic
> "pointers to variables": pointers are just references to
> data in memory (similar to regular variabes), because a
> command such as ptr_new(A) just duplicate the contents of A
> to a new memory location, so there is no such a thing as a
> true pointer to variable A, right?
>

Hi,

I don't know about the technical deep down stuff, but don't forget
that PTR_NEW accepts the /NO_COPY keyword. From the online help... "If
the NO_COPY keyword is set, the value data is taken away from the
InitExpr variable and attached directly to the heap variable. This
feature can be used to move data very efficiently. However, it has the
side effect of causing the InitExpr variable to become undefined."

Cheers,
Ben
