Subject: Re: renaming a variable without making a copy
Posted by David Fanning on Tue, 08 Dec 2009 13:18:02 GMT
View Forum Message <> Reply to Message

Dave Higgins writes:
.
> Can I rename a variable to a more apprорpriate name, instead of making
> a copy of the variable
>
> I'm using /OVERWRITE to save memory, through a number of operations on
> a large multi-dimensional array. I'd like to keep my code readable and
> rename the variable to reflect what the data currently are.

newName = Temporary(oldName)

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: renaming a variable without making a copy
Posted by David Higgins on Tue, 08 Dec 2009 14:51:18 GMT
View Forum Message <> Reply to Message

On 8 Dec, 13:18, David Fanning <n...@dfanning.com> wrote:
> Dave Higgins writes:
>
> .
>
>> Can I rename a variable to a more apprорpriate name, instead of making
>> a copy of the variable
>
>> I'm using /OVERWRITE to save memory, through a number of operations on
>> a large multi-dimensional array. I'd like to keep my code readable and
>> rename the variable to reflect what the data currently are.
>
> newName = Temporary(oldName)
>
> 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.")

Thanks!
Dave

Subject: Re: renaming a variable without making a copy
Posted by Kenneth P. Bowman on Tue, 08 Dec 2009 21:51:17 GMT
View Forum Message <> Reply to Message

In article <MPG.2587fff16b170da398968a@news.giganews.com>,
David Fanning <news@dfanning.com> wrote:

> newName = Temporary(oldName)

Can anyone explain to me what TEMPORARY actually does? The documentation says

> The TEMPORARY function returns a temporary copy of a variable, and sets
> the original variable to "undefined".

which makes no sense to me at all. Doesn't making a "temporary copy of a variable" occupy memory? Perhaps I am confused by the use of the name "TEMPORARY".

My concept of an IDL variable (which could easily be wrong) is: some metadata that describes the variable (what you get with the SIZE function) and the actual data that comprises the variable. These things could be in different places in memory, and the metadata could contain, for example, a pointer to the actual data. Most of the time, I don't need to know.

Does TEMPORARY wipe out the old metadata (replacing it with "undefined") and create new "unnamed" metadata that points to the data part of the destroyed variable?

The example in the Docs is not very revealing.

Why does

```
A = TEMPORARY(A) + 1
```

use less memory than

A = A + 1

??

I suppose there is a good reason that the latter example "creates a new
array for the result of the addition, places the sum into the new array,
assigns it to A, and then frees the old allocation of A", although it
just seems to me like the interpreter is being obtuse.

Cheers, Ken

Subject: Re: renaming a variable without making a copy
Posted by Michael Galloy on Tue, 08 Dec 2009 22:39:38 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman wrote:
> In article <MPG.2587fff16b170da398968a@news.giganews.com>,
> David Fanning <news@dfanning.com> wrote:
>
>> newName = Temporary(oldName)
>
> Can anyone explain to me what TEMPORARY actually does? The documentation
> says
>
> The TEMPORARY function returns a temporary copy of a variable, and sets
> the original variable to "undefined".
>
> which makes no sense to me at all. Doesn't making a "temporary copy
> of a variable" occupy memory? Perhaps I am confused by the use of the name
> "TEMPORARY".

TEMPORARY modifies the metadata of the variable passed to it, i.e., by
erasing its name and marking it as a temporary variable. All the other
metadata and the actual data can stay put.

> My concept of an IDL variable (which could easily be wrong) is: some
> metadata that describes the variable (what you get with the SIZE function)
> and the actual data that comprises the variable. These things could be
> in different places in memory, and the metadata could contain, for example,
> a pointer to the actual data. Most of the time, I don't need to know.

There's some other metadata and things depend on whether the variable is
an array or structure and what type it is, but this is basically correct.

> Does TEMPORARY wipe out the old metadata (replacing it with
> "undefined") and create new "unnamed" metadata that points to the data part
> of the destroyed variable?

>
> The example in the Docs is not very revealing.
>
> Why does
>
> A = TEMPORARY(A) + 1
>
> use less memory than
>
> A = A + 1
>
> ??
>
> I suppose there is a good reason that the latter example "creates a new
> array for the result of the addition, places the sum into the new array,
> assigns it to A, and then frees the old allocation of A", although it
> just seems to me like the interpreter is being obtuse.

The interpreter probably could look for these things and optimize, but
generally when it sees an expression like A + 1 it must create a
temporary variable to hold the result. Only later does it see that this
will be assigned back to A.

IDL can handle temporary variables differently. It knows the result of
the expression

  tempVar + 1

can be placed right back into the temporary variable. It also knows that
when assigning a temporary variable to a new variable name, as in

  A = temporary(A) + 1

the "assignment" can just attached the name "A" and change the metadata
of the variable to indicate that it is no longer a temporary variable.

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: renaming a variable without making a copy
Posted by David Fanning on Tue, 08 Dec 2009 22:39:40 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman writes:

> In article <MPG.2587fff16b170da398968a@news.giganews.com>,
> David Fanning <news@dfanning.com> wrote:
>
>> newName = Temporary(oldName)
>
> Can anyone explain to me what TEMPORARY actually does? The documentation
> says
>
> The TEMPORARY function returns a temporary copy of a variable, and sets
> the original variable to "undefined".
>
> which makes no sense to me at all. Doesn't making a "temporary copy
> of a variable" occupy memory? Perhaps I am confused by the use of the name
> "TEMPORARY".
>
> My concept of an IDL variable (which could easily be wrong) is: some
> metadata that describes the variable (what you get with the SIZE function)
> and the actual data that comprises the variable. These things could be
> in different places in memory, and the metadata could contain, for example,
> a pointer to the actual data. Most of the time, I don't need to know.
>
> Does TEMPORARY wipe out the old metadata (replacing it with
> "undefined") and create new "unnamed" metadata that points to the data part
> of the destroyed variable?
>
> The example in the Docs is not very revealing.

Here is how I wave my hands around this when explaining it
in an IDL class. Remember, I am speaking metaphorically here.
I have *no* idea what actually happens. ;-)

You are right, a variable in IDL is composed of some metadata,
one part of which is the variable's name, and some machine
memory, where the variable lives. I like to say the variable
is "attached" to the machine memory. When you issue a command
like this:

```
newVar = Temporary(oldVar) + 1
```

You are saying to IDL, "Take that machine memory that is attached
to oldVar and temporarily use it to perform whatever operation
you are doing." Then, when you are finished, make another variable,
newVar, and attach this temporary memory permanently to this variable.
In IDL there is a rule that only one variable at a time can be
permanently attached to machine memory, so the act of attaching this
memory to newVar is to remove it from oldVar. A variable that has
no machine memory attached to it is, by definition, an undefined

variable.

> Why does
>
> A = TEMPORARY(A) + 1
>
> use less memory than
>
> A = A + 1
>
> I suppose there is a good reason that the latter example "creates a new
> array for the result of the addition, places the sum into the new array,
> assigns it to A, and then frees the old allocation of A", although it
> just seems to me like the interpreter is being obtuse.

I'm sure there is a good reason. And if I think about it long
enough, I'm sure it will come to me. Meantime, you may have
to take it on faith that IDL just works that way. :-)

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: renaming a variable without making a copy
Posted by lecacheux.alain on Wed, 09 Dec 2009 09:42:42 GMT
View Forum Message <> Reply to Message

On 8 déc, 23:39, David Fanning <n...@dfanning.com> wrote:
> Kenneth P. Bowman writes:
>> In article <MPG.2587fff16b170da3989...@news.giganews.com>,
>> David Fanning <n...@dfanning.com> wrote:
>
>>> newName = Temporary(oldName)
>
>> Can anyone explain to me what TEMPORARY actually does? The documentation
>> says
>
>> The TEMPORARY function returns a temporary copy of a variable, and sets
>> the original variable to "undefined".
>
>> which makes no sense to me at all. Doesn't making a "temporary copy

>> of a variable" occupy memory? Perhaps I am confused by the use of the name
>> "TEMPORARY".
>
>> My concept of an IDL variable (which could easily be wrong) is: some
>> metadata that describes the variable (what you get with the SIZE function)
>> and the actual data that comprises the variable. These things could be
>> in different places in memory, and the metadata could contain, for example,
>> a pointer to the actual data. Most of the time, I don't need to know.
>
>> Does TEMPORARY wipe out the old metadata (replacing it with
>> "undefined") and create new "unnamed" metadata that points to the data part
>> of the destroyed variable?
>
>> The example in the Docs is not very revealing.
>
> Here is how I wave my hands around this when explaining it
> in an IDL class. Remember, I am speaking metaphorically here.
> I have *no* idea what actually happens. ;-)
>
> You are right, a variable in IDL is composed of some metadata,
> one part of which is the variable's name, and some machine
> memory, where the variable lives. I like to say the variable
> is "attached" to the machine memory. When you issue a command
> like this:
>
> newVar = Temporary(oldVar) + 1
>
> You are saying to IDL, "Take that machine memory that is attached
> to oldVar and temporarily use it to perform whatever operation
> you are doing." Then, when you are finished, make another variable,
> newVar, and attach this temporary memory permanently to this variable.
> In IDL there is a rule that only one variable at a time can be
> permanently attached to machine memory, so the act of attaching this
> memory to newVar is to remove it from oldVar. A variable that has
> no machine memory attached to it is, by definition, an undefined
> variable.
>
>> Why does
>
>> A = TEMPORARY(A) + 1
>
>> use less memory than
>
>> A = A + 1
>
>> I suppose there is a good reason that the latter example "creates a new
>> array for the result of the addition, places the sum into the new array,
>> assigns it to A, and then frees the old allocation of A", although it

>> just seems to me like the interpreter is being obtuse.
>
> I'm sure there is a good reason. And if I think about it long
> enough, I'm sure it will come to me. Meantime, you may have
> to take it on faith that IDL just works that way. :-)
>
> 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.")- Masquer le texte des messages précédents -
>

Here is a question, maybe related to the previous one, regarding
variable typecast in IDL.
Changing the type of one scalar variable, even a vector, is fairly
easy by using the FIX function.
But let suppose that I get a variable (for instance from an external
routine, by reading a shared memory,
a socket, etc...) that is described as a vector of byte (or integer or
float or anything else).
I want to further consider this variable as a (elsewhere defined)
structure (in the IDL sense).
In other words, I want to typecast an untyped variable to a structured
one : what is the way in IDL ?

Subject: Re: renaming a variable without making a copy
Posted by David Fanning on Wed, 09 Dec 2009 13:19:08 GMT
View Forum Message <> Reply to Message

alx writes:

.
> Here is a question, maybe related to the previous one, regarding
> variable typecast in IDL.
> Changing the type of one scalar variable, even a vector, is fairly
> easy by using the FIX function.
> But let suppose that I get a variable (for instance from an external
> routine, by reading a shared memory,
> a socket, etc...) that is described as a vector of byte (or integer or
> float or anything else).
> I want to further consider this variable as a (elsewhere defined)
> structure (in the IDL sense).

> In other words, I want to typecast an untyped variable to a structured
> one : what is the way in IDL ?

There is no general way. Structures often have "hidden"
information inside them, mostly to align fields of data
with machine words, etc. Unless you know *exactly* how
a structure is built in whatever language you are dealing
with, it will be difficult to convert a stream of bytes
back into a structure on the IDL side of things. I've
heard of it being done, but I think it is fraught will
peril. :-)

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: renaming a variable without making a copy
Posted by lecacheux.alain on Wed, 09 Dec 2009 14:33:34 GMT
View Forum Message <> Reply to Message

On 9 déc, 14:19, David Fanning <n...@dfanning.com> wrote:

> ... Unless you know *exactly* how
> a structure is built in whatever language you are dealing
> with, it will be difficult to convert a stream of bytes
> back into a structure on the IDL side of things. > --

But IDL provides this facility with files ! IDL structures are written
to and read from binary files,
independently of C padding (cf. structure 'length' and 'data_length').
It would be nice to extend
this possibility to a stream of bytes and, even more usefully, to
pointers ...
Unfortunately, taking into account a valid pointer, I did not yet
discover the way, in IDL, to change the
underlying variable.
alx.

Subject: Re: renaming a variable without making a copy
Posted by Kenneth P. Bowman on Wed, 09 Dec 2009 15:02:39 GMT
View Forum Message <> Reply to Message

In article <hfmki3$1tf$1@aioe.org>, mgalloy <mgalloy@gmail.com> wrote:

> IDL can handle temporary variables differently. It knows the result of
> the expression
>
>   tempVar + 1
>
> can be placed right back into the temporary variable. It also knows that
> when assigning a temporary variable to a new variable name, as in
>
>   A = temporary(A) + 1
>
> the "assignment" can just attached the name "A" and change the metadata
> of the variable to indicate that it is no longer a temporary variable.

Thanks, Mike. After reading your post I found the Temporary Variables
page in the IDL internals documentation, which is actually relatively
clear:

"IDL maintains a pool of nameless variables known as temporary variables.
These variables are used by the interpreter to hold temporary results
from evaluating expressions, and are also used within system procedures
and functions that need temporary workspace. In addition, system
functions often obtain a temporary variable to return the result of
their operation to the interpreter."

Ken

Subject: Re: renaming a variable without making a copy
Posted by R.Bauer on Wed, 09 Dec 2009 17:06:05 GMT
View Forum Message <> Reply to Message

>
> Here is a question, maybe related to the previous one, regarding
> variable typecast in IDL.
> Changing the type of one scalar variable, even a vector, is fairly
> easy by using the FIX function.
> But let suppose that I get a variable (for instance from an external
> routine, by reading a shared memory,
> a socket, etc...) that is described as a vector of byte (or integer or
> float or anything else).
> I want to further consider this variable as a (elsewhere defined)
> structure (in the IDL sense).

> In other words, I want to typecast an untyped variable to a structured
> one : what is the way in IDL ?

read about reads or execute

cheers
Reimar
