
Subject: IDL Save Object Resoration

Posted by [David B](#) on Tue, 21 Apr 2015 11:17:01 GMT

[View Forum Message](#) <> [Reply to Message](#)

Suppose I create a variable:

```
-----  
;Create arrays  
a = [1,2,3,4,5,6]  
b =[7,8,9,10]  
c = INDGEN(10,10)  
  
;Save these arrays normally  
save, a, b, c, filename = 'file.sav'  
  
;Run a reset  
.reset  
  
;Restore these variables, but into an object  
sobj= OBJ_NEW('IDL_Savefile', 'file.sav')  
  
;Extract names  
names = sObj->Names()  
  
;Print  
Print, names  
  
-----
```

The problem is that I need the variables to be named something differently, but I cannot extract the variables into a new name. Non of my objects are heap variables, just standard variables.

For example, following this:

[http://www.exelisvis.com/docs/idl_savefile__restore.html#obj ects_misc_904195448_1034949](http://www.exelisvis.com/docs/idl_savefile__restore.html#obj%20ects_misc_904195448_1034949)

One can restore Pointers and Objects into a new object/pointer, so you can automatically rename your variables on the fly with the line:

```
-----  
; Restore the heap variable, associating it with a new regular  
; variable. Note that ptrName is (in this case) a one-element array.  
sObj->Restore, ptrName[0], /POINTER_HEAPVAR, NEW_HEAPVAR=myNewPtr  
-----
```

I have an almost solid reason for doing this in my case. What I am therefore saying is that I want for example (but I cannot do this, because 'new_variable' is not an option in the restore method!)

```
FOR i = 0, n_elements(names)-1, 1 DO BEGIN
```

```
    subj->Restore, names[i], new_variable = 'new_'+names[i]
```

```
ENDFOR
```

SO I end up with the following as a result:

```
new_a = a
```

```
new_b = b
```

```
new_c = c
```

Where I can restore a simple variable, like an array, into a new name. I may be missing the point here but I cannot think of a way to do this in a strait forward way, and the 'EXECUTE' command is out of the question for these objects.

Clearly I am being stupid, but I really am stuck. Also, I know this method does not work as:

```
new = obj->Restore, names[i], new_variable = 'new_'+names[i]
```

otherwise, my problem would be solved easily. Can anyone offer insight into this and point out my mistake?

Thanks much

David

Subject: Re: IDL Save Object Resoration

Posted by [Michael Galloy](#) on Tue, 21 Apr 2015 12:55:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 4/21/15 5:17 AM, David B wrote:

> Suppose I create a variable:

>

> ----- ;Create arrays a =

> [1,2,3,4,5,6] b =[7,8,9,10] c = INDGEN(10,10)

>

> ;Save these arrays normally save, a, b, c, filename = 'file.sav'

>

> ;Run a reset ..reset

>

> ;Restore these variables, but into an object subj=

> OBJ_NEW('IDL_Savefile', 'file.sav')

>

> ;Extract names names = sObj->Names()

>

> ;Print Print, names

>
> -----
>
> The problem is that I need the variables to be named something
> differently, but I cannot extract the variables into a new name. Non
> of my objects are heap variables, just standard variables.
>
> For example, following this:
>
> http://www.exelisvis.com/docs/idl_savefile__restore.html#objects_misc_904195448_1034949
>
> One can restore Pointers and Objects into a new object/pointer, so
> you can automatically rename your variables on the fly with the
> line:
>
> ----- ; Restore the heap variable,
> associating it with a new regular ; variable. Note that ptrName is
> (in this case) a one-element array. sObj->Restore, ptrName[0],
> /POINTER_HEAPVAR, NEW_HEAPVAR=myNewPtr
> -----
>
> I have an almost solid reason for doing this in my case. What I am
> therefore saying is that I want for example (but I cannot do this,
> because 'new_variable' is not an option in the restore method!)
>
> FOR i = 0, n_elements(names)-1, 1 DO BEGIN
>
> sobj->Restore, names[i], new_variable = 'new_'+names[i]
>
> ENDFOR
>
> SO I end up with the following as a result:
>
> new_a = a new_b = b new_c = c
>
> Where I can restore a simple variable, like an array, into a new
> name. I may be missing the point here but I cannot think of a way to
> do this in a strait forward way, and the 'EXECUTE' command is out of
> the question for these objects.
>
> Clearly I am being stupid, but I really am stuck. Also, I know this
> method does not work as:
>
> new = obj->Restore, names[i], new_variable = 'new_'+names[i]
>
> otherwise, my problem would be solved easily. Can anyone offer
> insight into this and point out my mistake?
>

> Thanks much
>
> David
>

I have a routine MG_SAVE_GETDATA that can do this:

https://github.com/mgalloy/mglib/blob/master/src/save/mg_save_getdata.pro

Just call it like:

```
cow_filename = file_which('cow10.sav')  
my_name_for_polylist = mg_save_getdata(cow_filename, 'polylist')
```

Mike

--

Michael Galloy

www.michaelgalloy.com

Modern IDL: A Guide to IDL Programming (<http://modernidl.idldev.com>)

Subject: Re: IDL Save Object Resoration

Posted by [David B](#) on Tue, 21 Apr 2015 13:01:40 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thanks Micheal,

I shall have a look at this. It seems a little bit of an obvious trick to miss out on the key wording angle.

Regards

David

Subject: Re: IDL Save Object Resoration

Posted by [Jim Pendleton](#) on Tue, 21 Apr 2015 13:01:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Tuesday, April 21, 2015 at 5:17:04 AM UTC-6, David B wrote:

> Suppose I create a variable:

>

> -----

> ;Create arrays

> a = [1,2,3,4,5,6]

> b = [7,8,9,10]

> c = INDGEN(10,10)

```

>
> ;Save these arrays normally
> save, a, b, c, filename = 'file.sav'
>
> ;Run a reset
> .reset
>
> ;Restore these variables, but into an object
> sobj= OBJ_NEW('IDL_Savefile', 'file.sav')
>
> ;Extract names
> names = sobj->Names()
>
> ;Print
> Print, names
>
> -----
>
> The problem is that I need the variables to be named something differently, but I cannot extract
the variables into a new name. Non of my objects are heap variables, just standard variables.
>
> For example, following this:
>
> http://www.exelisvis.com/docs/idl\_savefile\_\_restore.html#obj\_ects\_misc\_904195448\_1034949
>
> One can restore Pointers and Objects into a new object/pointer, so you can automatically
rename your variables on the fly with the line:
>
> -----
> ; Restore the heap variable, associating it with a new regular
> ; variable. Note that ptrName is (in this case) a one-element array.
> sobj->Restore, ptrName[0], /POINTER_HEAPVAR, NEW_HEAPVAR=myNewPtr
> -----
>
> I have an almost solid reason for doing this in my case. What I am therefore saying is that I
want for example (but I cannot do this, because 'new_variable' is not an option in the restore
method!)
>
> FOR i = 0, n_elements(names)-1, 1 DO BEGIN
>
>     sobj->Restore, names[i], new_variable = 'new_'+names[i]
>
> ENDFOR
>
> SO I end up with the following as a result:
>
> new_a = a
> new_b = b

```

```
> new_c = c
>
> Where I can restore a simple variable, like an array, into a new name. I may be missing the
point here but I cannot think of a way to do this in a strait forward way, and the 'EXECUTE'
command is out of the question for these objects.
>
> Clearly I am being stupid, but I really am stuck. Also, I know this method does not work as:
>
> new = obj->Restore, names[i], new_variable = 'new_'+names[i]
>
> otherwise, my problem would be solved easily. Can anyone offer insight into this and point out
my mistake?
>
> Thanks much
>
> David
```

Why not wrap your Restore call in a procedure and use the names you want as parameter or keyword arguments?

```
pro myrestore, sObj, a = a, b = b, ...
sobj->restore, a, ...
end

pro mymain
sobj = idl_savefile(...)
myrestore, sobj, a = new_a, b = new_b, ...
help, new_a, new_b, ...
end
```

If you're going to generate the new names on the fly, you might want to consider using a hash instead.

Jim P.

Subject: Re: IDL Save Object Resoration
Posted by [David B](#) on Tue, 21 Apr 2015 13:56:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tuesday, April 21, 2015 at 2:01:58 PM UTC+1, Jim P wrote:
> On Tuesday, April 21, 2015 at 5:17:04 AM UTC-6, David B wrote:
>> Suppose I create a variable:
>>
>> -----
>> ;Create arrays
>> a = [1,2,3,4,5,6]
>> b = [7,8,9,10]

```

>> c = INDGEN(10,10)
>>
>> ;Save these arrays normally
>> save, a, b, c, filename = 'file.sav'
>>
>> ;Run a reset
>> .reset
>>
>> ;Restore these variables, but into an object
>> sobj= OBJ_NEW('IDL_Savefile', 'file.sav')
>>
>> ;Extract names
>> names = sobj->Names()
>>
>> ;Print
>> Print, names
>>
>> -----
>>
>> The problem is that I need the variables to be named something differently, but I cannot
>> extract the variables into a new name. None of my objects are heap variables, just standard
>> variables.
>>
>> For example, following this:
>>
>> http://www.exelisvis.com/docs/idl\_savefile\_\_restore.html#obj
>> ects\_misc\_904195448\_1034949
>>
>> One can restore Pointers and Objects into a new object/pointer, so you can automatically
>> rename your variables on the fly with the line:
>>
>> -----
>> ; Restore the heap variable, associating it with a new regular
>> ; variable. Note that ptrName is (in this case) a one-element array.
>> sobj->Restore, ptrName[0], /POINTER_HEAPVAR, NEW_HEAPVAR=myNewPtr
>> -----
>>
>> I have an almost solid reason for doing this in my case. What I am therefore saying is that I
>> want for example (but I cannot do this, because 'new_variable' is not an option in the restore
>> method!)
>>
>> FOR i = 0, n_elements(names)-1, 1 DO BEGIN
>>
>>     sobj->Restore, names[i], new_variable = 'new_'+names[i]
>>
>> ENDFOR
>>
>> SO I end up with the following as a result:

```

```

>>
>> new_a = a
>> new_b = b
>> new_c = c
>>
>> Where I can restore a simple variable, like an array, into a new name. I may be missing the
point here but I cannot think of a way to do this in a strait forward way, and the 'EXECUTE'
command is out of the question for these objects.
>>
>> Clearly I am being stupid, but I really am stuck. Also, I know this method does not work as:
>>
>> new = obj->Restore, names[i], new_variable = 'new_'+names[i]
>>
>> otherwise, my problem would be solved easily. Can anyone offer insight into this and point out
my mistake?
>>
>> Thanks much
>>
>> David
>
> Why not wrap your Restore call in a procedure and use the names you want as parameter or
keyword arguments?
>
> pro myrestore, sObj, a = a, b = b, ...
> sobj->restore, a, ...
> end
>
> pro mymain
> sobj = idl_savefile(...)
> myrestore, sobj, a = new_a, b = new_b, ...
> help, new_a, new_b, ...
> end
>
> If you're going to generate the new names on the fly, you might want to consider using a hash
instead.
>
> Jim P.

```

I THINK that is also a valid approach too! A further possibility is such that:

```

-----
;Recover the names of the objects
name = sObj->Names()

```

```

;Name is then a STRARR ---> name = ['a', 'b', 'c']

```

```

;Then do a loop
FOR i = 0, n_elements(name)-1, +1 DO BEGIN

```

```
res = EXECUTE('new_'+name[jj]+' = TEMPORARY(name[jj])'
```

```
ENDFOR
```

The Hash method is also a good approach from Michael and works fine!

Regards

David

Subject: Re: IDL Save Object Resoration

Posted by [Michael Galloy](#) on Tue, 21 Apr 2015 20:45:15 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 4/21/15 7:56 AM, David B wrote:

> On Tuesday, April 21, 2015 at 2:01:58 PM UTC+1, Jim P wrote:

>> On Tuesday, April 21, 2015 at 5:17:04 AM UTC-6, David B wrote:

>>> Suppose I create a variable:

>>>

>>> ----- ;Create arrays a =

>>> [1,2,3,4,5,6] b=[7,8,9,10] c = INDGEN(10,10)

>>>

>>> ;Save these arrays normally save, a, b, c, filename = 'file.sav'

>>>

>>> ;Run a reset .reset

>>>

>>> ;Restore these variables, but into an object sobj=

>>> OBJ_NEW('IDL_Savefile', 'file.sav')

>>>

>>> ;Extract names names = sObj->Names()

>>>

>>> ;Print Print, names

>>>

>>> -----

>>>

>>> The problem is that I need the variables to be named something

>>> differently, but I cannot extract the variables into a new name.

>>> Non of my objects are heap variables, just standard variables.

>>>

>>> For example, following this:

>>>

>>> [http://www.exelisvis.com/docs/idl_savefile__restore.html#obj](http://www.exelisvis.com/docs/idl_savefile__restore.html#objects_misc_904195448_1034949)
[ects_misc_904195448_1034949](http://www.exelisvis.com/docs/idl_savefile__restore.html#objects_misc_904195448_1034949)

>>>

>>>

```

>>>
One can restore Pointers and Objects into a new object/pointer, so you
can automatically rename your variables on the fly with the line:
>>>
>>> ----- ; Restore the heap
>>> variable, associating it with a new regular ; variable. Note that
>>> ptrName is (in this case) a one-element array. sObj->Restore,
>>> ptrName[0], /POINTER_HEAPVAR, NEW_HEAPVAR=myNewPtr
>>> -----
>>>
>>> I have an almost solid reason for doing this in my case. What I
>>> am therefore saying is that I want for example (but I cannot do
>>> this, because 'new_variable' is not an option in the restore
>>> method!)
>>>
>>> FOR i = 0, n_elements(names)-1, 1 DO BEGIN
>>>
>>> subj->Restore, names[i], new_variable = 'new_'+names[i]
>>>
>>> ENDFOR
>>>
>>> SO I end up with the following as a result:
>>>
>>> new_a = a new_b = b new_c = c
>>>
>>> Where I can restore a simple variable, like an array, into a new
>>> name. I may be missing the point here but I cannot think of a way
>>> to do this in a strait forward way, and the 'EXECUTE' command is
>>> out of the question for these objects.
>>>
>>> Clearly I am being stupid, but I really am stuck. Also, I know
>>> this method does not work as:
>>>
>>> new = obj->Restore, names[i], new_variable = 'new_'+names[i]
>>>
>>> otherwise, my problem would be solved easily. Can anyone offer
>>> insight into this and point out my mistake?
>>>
>>> Thanks much
>>>
>>> David
>>
>> Why not wrap your Restore call in a procedure and use the names you
>> want as parameter or keyword arguments?
>>
>> pro myrestore, sObj, a = a, b = b, ... subj->restore, a, ... end
>>
>> pro mymain subj = idl_savefile(...) myrestore, subj, a = new_a, b =

```

```

>> new_b, ... help, new_a, new_b, ... end
>>
>> If you're going to generate the new names on the fly, you might
>> want to consider using a hash instead.
>>
>> Jim P.
>
> I THINK that is also a valid approach too! A further possibility is
> such that:
>
> ----- ;Recover the names of the
> objects name = sObj->Names()
>
> ;Name is then a STRARR ---> name = ['a', 'b', 'c']
>
> ;Then do a loop FOR i = 0, n_elements(name)-1, +1 DO BEGIN
>
> res = EXECUTE('new_'+name[jj]+' = TEMPORARY(name[jj])'
>
> ENDFOR -----
>
> The Hash method is also a good approach from Michael and works fine!

```

Jim's method is essentially what I am doing, just returning one variable at a time instead of using keywords. But the hash actually seems like a pretty good way to do it if you have multiple values that you want back at once. I think I will add an /ALL keyword that puts them all in a hash to return.

-Mike

Subject: Re: IDL Save Object Resoration

Posted by [Michael Galloy](#) on Tue, 21 Apr 2015 20:48:01 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 4/21/15 2:45 PM, Michael Galloy wrote:

```

> Jim's method is essentially what I am doing, just returning one variable
> at a time instead of using keywords. But the hash actually seems like a
> pretty good way to do it if you have multiple values that you want back
> at once. I think I will add an /ALL keyword that puts them all in a hash
> to return.
>
> -Mike

```

Ah, sorry, I'm one step ahead of myself! I see what you mean with the hash now. I already do that if no variable name is passed in. Present me is impressed with the forward thinking of past me.

-Mike

Subject: Re: IDL Save Object Resoration

Posted by [Jim Pendleton](#) on Wed, 22 Apr 2015 02:33:15 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Tuesday, April 21, 2015 at 2:47:41 PM UTC-6, Mike Galloy wrote:

> On 4/21/15 2:45 PM, Michael Galloy wrote:

>> Jim's method is essentially what I am doing, just returning one variable
>> at a time instead of using keywords. But the hash actually seems like a
>> pretty good way to do it if you have multiple values that you want back
>> at once. I think I will add an /ALL keyword that puts them all in a hash
>> to return.

>>

>> -Mike

>

> Ah, sorry, I'm one step ahead of myself! I see what you mean with the
> hash now. I already do that if no variable name is passed in. Present me
> is impressed with the forward thinking of past me.

>

> -Mike

Sorry, Mike. Didn't see your post until I'd posted mine. I think your time machine is on the fritz again. Just like it was next year.

Jim P.
