Subject: Copying a hash
Posted by Matt[3] on Mon, 06 Aug 2012 20:54:01 GMT
View Forum Message <> Reply to Message

Hi All,

Does anyone know if there's a simple way that I can make a copy of a hash, which I can then edit independently of the original? For example, it seems that, like a pointer, changes that I make to the copy are also applied to the original:

```
IDL> original=hash('A', [1, 2])
IDL> copy=original
IDL> copy['A', 1]=10
IDL> print, copy
A:       1      10
IDL> print, original
A:       1      10
```

I can copy to a new hash key-by-key:

```
copy=hash()
foreach variable, original, key do copy[key]=original[key]
```

Which works fine, unless one of the elements in the hash is itself a hash, then I end up with the same problem one level down.

Is there something simple I'm missing here?

Cheers,

Matt

Subject: Re: Copying a hash
Posted by David Fanning on Mon, 06 Aug 2012 23:41:12 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> Bummer. To be honest, I'm not sure what the correct behaviour should be. Recursively copy all the components? I guess if
> we think of the numbers and strings as objects also, then the answer should probably be yes.... ? Why duplicate one type
> of object (int, float, or string) but not another (hash or list)? Still... it just doesn't seem right.

I think this takes us back to the need for a "deep copy" in objects.

http://www.idlcoyote.com/tips/copy_objects.html

But, we have only been requesting it for 9 years, I see by
the date on the article. I think the standard is 12 years
before they either fix the problem or consign the requester
to the loony bin. :-)

Cheers,

David

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

Subject: Re: Copying a hash
Posted by Matt[3] on Tue, 07 Aug 2012 15:44:45 GMT
View Forum Message <> Reply to Message

On Monday, August 6, 2012 7:41:12 PM UTC-4, David Fanning wrote:
> Paul van Delst writes:
>
>
>
>> Bummer. To be honest, I'm not sure what the correct behaviour should be. Recursively copy all the components? I guess if
>
>> we think of the numbers and strings as objects also, then the answer should probably be yes.... ? Why duplicate one type
>
>> of object (int, float, or string) but not another (hash or list)? Still... it just doesn't seem right.
>
>
>
> I think this takes us back to the need for a "deep copy" in objects.
>
>
>
> http://www.idlcoyote.com/tips/copy_objects.html
>
>
>
> But, we have only been requesting it for 9 years, I see by
>
> the date on the article. I think the standard is 12 years
>

> before they either fix the problem or consign the requester
>
> to the loony bin. :-)
>
>
>
> Cheers,
>
>
>
> David
>
>
>
> --
>
> David Fanning, Ph.D.
>
> Fanning Software Consulting, Inc.
>
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Hi Guys,

Thanks for the help, and sorry for somehow missing that rather clear bit of documentation.

Yes, I'm not sure what the behavior should be either when there is a hash within a hash. The default behavior seems likely to cause trouble! Anyway, to copy down through one or two hash levels, the following lines seem to work:

```
copy=hash()
foreach variable, original, key do copy[key]=original[key, *]
```

I'm sure there's a smart way of doing this recursively for an indefinite number of levels, but this works for me, for now.

Cheers,

Matt

Subject: Re: Copying a hash
Posted by Matt[3] on Thu, 09 Aug 2012 18:25:51 GMT
View Forum Message <> Reply to Message

On Tuesday, August 7, 2012 11:44:45 AM UTC-4, Matt wrote:

> On Monday, August 6, 2012 7:41:12 PM UTC-4, David Fanning wrote:
>
>> Paul van Delst writes:
>
>>
>
>>
>
>>
>

>>> Bummer. To be honest, I'm not sure what the correct behaviour should be. Recursively copy all the components? I guess if
>
>>
>

>>> we think of the numbers and strings as objects also, then the answer should probably be yes.... ? Why duplicate one type
>
>>
>

>>> of object (int, float, or string) but not another (hash or list)? Still... it just doesn't seem right.
>
>>
>
>>
>
>>
>

>> I think this takes us back to the need for a "deep copy" in objects.
>
>>
>
>>
>
>>
>

>> http://www.idlcoyote.com/tips/copy_objects.html
>
>>
>
>>
>
>>
>

>> But, we have only been requesting it for 9 years, I see by
>
>>
>

>> the date on the article. I think the standard is 12 years
>
>>
>
>> before they either fix the problem or consign the requester
>
>>
>
>> to the loony bin. :-)
>
>>
>
>>
>
>>
>
>>
>
>> Cheers,
>
>>
>
>>
>
>>
>
>>
>
>> David
>
>>
>
>>
>
>>
>
>>
>
>> --
>
>>
>
>> David Fanning, Ph.D.
>
>>
>
>> Fanning Software Consulting, Inc.
>
>>
>
>> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
>>
>

>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")
>
>
>
> Hi Guys,
>
>
>
> Thanks for the help, and sorry for somehow missing that rather clear bit of documentation.
>
>
>
> Yes, I'm not sure what the behavior should be either when there is a hash within a hash. The default behavior seems likely to cause trouble! Anyway, to copy down through one or two hash levels, the following lines seem to work:
>
>
>
> copy=hash()
>
> foreach variable, original, key do copy[key]=original[key, *]
>
>
>
> I'm sure there's a smart way of doing this recursively for an indefinite number of levels, but this works for me, for now.
>
>
>
> Cheers,
>
>
>
> Matt

Probably no one cares about this but me, but the code I posted above doesn't copy arrays stored in a hash correctly (it turns multi-dimensional arrays into 1-D arrays). This works though:

```
copy=hash()
foreach variable, original, key do begin
  if typeName(original[key]) eq 'HASH' then begin
    copy[key]=original[key, *]
  endif else begin
    copy[key]=original[key]
  endelse
endforeach
```

Subject: Re: Copying a hash
Posted by Matt Francis on Fri, 10 Aug 2012 00:44:33 GMT
View Forum Message <> Reply to Message

There is actually nothing wrong with not providing a 'deep copy' functionality. In most (all?) languages with full object oriented programming support (which I don't include IDL in yet, for the lack of several key features) it is always the responsibility of the coder to provide a copy constructor.

I do this for all the IDL custom objects I create by considering it mandatory to implement a copy function that returns an instance of the copied object. That's no different from what is required in genuine OO languages.

Subject: Re: Copying a hash
Posted by Bob[4] on Fri, 17 Aug 2012 22:25:44 GMT
View Forum Message <> Reply to Message

On Thursday, August 9, 2012 6:44:33 PM UTC-6, Bogdanovist wrote:
> There is actually nothing wrong with not providing a 'deep copy' functionality. In most (all?) languages with full object oriented programming support (which I don't include IDL in yet, for the lack of several key features) it is always the responsibility of the coder to provide a copy constructor.
>
>
>
> I do this for all the IDL custom objects I create by considering it mandatory to implement a copy function that returns an instance of the copied object. That's no different from what is required in genuine OO languages.

I perhaps agree with you sentiment for user defined objects. But the hash object is defined in IDL internals so it would be nice if they added a deep copy function to it.