
Subject: Re: saving variables between calls to a procedure?
Posted by [Paul Van Delst\[1\]](#) on Thu, 01 Aug 2002 14:45:52 GMT
[View Forum Message](#) <> [Reply to Message](#)

Mark Hadfield wrote:

```
>
> "David Fanning" <david@dfanning.com> wrote in message
> news:MPG.17b21214ff7d9b35989944@news.frii.com...
>
>> Paul van Delst (paul.vandelst@noaa.gov) writes:
>>
>>>> pro define,ptr
>>>>   *ptr=[*ptr,10]
>>>> end
>>>
>>> Hmm. That seems like an extremely dangerous thing to do - couldn't
>>> you clobber something by concatenating like that? If IDL is smart
>>> enough to recognise that the next bit of memory may be used by
>>> something else it then seems that you would end up with a
>>> non-contiguous data structure (in the figurative).
>>
>> This doesn't seem dangerous to me (perhaps because I use the
>> construct all the time). It seems like one of those wonderful things
>> IDL occasionally does that makes you think to yourself "Now, by God,
>> that's how software *ought* to work!"
>>
>> In any case, it works, over and over and over. And it never occurred
>> to me that non-contiguous data storage could be involved, even
>> remotely.
>
> Extending an array a with the a = [a,b] syntax doesn't create a
> non-contiguous data structure.
```

I'm not worried about extending arrays like `a=[a,b]`, but pointers ala `*ptr=[*ptr,10]`. If `ptr` is pointing to some data structure (scalar or array), it seems feasible to me that the next position in memory could be occupied by something else. If one then does a `*ptr=[*ptr,10]`, what happens to what was in that memory? Is it "protected" somehow so that the value "10" is then placed in the next "free" spot in memory giving a non-contiguous array [not efficient, but safe], or are the contents clobbered and replaced by the value "10" [very very bad], or are the contents moved to some other memory location so that the original position can be used to store the value "10" [relatively efficient(?) and safe].

```
> What is does is create a new array a,
> insert the elements from the existing a and b into it, then delete the
> old a.
```

So does this mean that

```
*ptr = [*ptr,10]
```

is equivalent to:

```
a = *ptr  
ptr_free, ptr  
a=[a,10]  
ptr=ptr_new(temporary(a))
```

?

(Which was my original question, just poorly phrased)

paulv

--

Paul van Delst	
CIMSS @ NOAA/NCEP/EMC	Beer is good.
Ph: (301)763-8000 x7274	My wife.
Fax:(301)763-8545	