Subject: Re: Modifying an array while conserving memory
Posted by R.Bauer on Fri, 24 May 2002 09:49:14 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:
>
>> Why not using pointer:
>>
>> ptr1 = PTR_NEW(FINDGEN(1000))
>> insert = PTR_NEW(RANDOMU(seed,100))
>> a = PTR_NEW([(*ptr1)[0:499], (*insert), (*ptr1)[500:*]])
>>
>>
>> HELP,*a
>
> The problem with using pointers as above is that you are not actually
> using the pointer, but copying the data contained within. Take a look at
> the heap after doing the above:
>
> IDL> help, /heap
> Heap Variables:
> # Pointer: 3
> # Object : 0
>
> <PtrHeapVar1> FLOAT = Array[1000]
> <PtrHeapVar2> FLOAT = Array[100]
> <PtrHeapVar3> FLOAT = Array[1100]
>
> This shows that until I physically free the pointers 'ptr1' and 'insert',
> I have used exactly double the memory as I now have a copy of each
> variable.
>
> Rather than inserting the data into the middle, I would (at this point) be
> happy enough just concatenating to arrays...
>
> IDL> ptr1 = PTR_NEW(FINDGEN(1000))
> IDL> ptr2 = PTR_NEW(RANDOMU(seed,100))
> IDL> a = [ptr1,ptr2]
> IDL> print, *a ; fails
> IDL> print, *(a) ; fails
> IDL> print, *a(*) ; fails
> IDL> print, *a[0] ; prints findgen(1000) (i.e. not what I want)
> IDL> print, *(a)(*) ; fails... Score: IDL 5 ; Randall 0

If you use my dref function

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/dref.tar.gz

; PURPOSE:
; This functions derefences pointers. If last dimension is 1 this is returned and not the
; standard IDL Array without last dimension 1.
; If value is a array of pointer the values are concatinated by concatinate_arrays at the last dimension
; If value isn't a pointer this value is returned but with the right dimensions.

e.g.
IDL> help,dref(a)
<Expression>   FLOAT   = Array[1100]

e.g.:
IDL> help,dref(a,/free)
<Expression>   FLOAT   = Array[1100]

; free means ptr_free

this is solved but during operation the memory is double times allocated.

regards
Reimar

>
> Because IDL doesn't keep track of what type of data is in a pointer, the
> above is protecting me from doing silly things:
>
> IDL> ptr1 = PTR_NEW(FINDGEN(1000))
> IDL> insert = ptr_new('test')
> IDL> a = [ptr1, insert]
>
> Perhaps this is something for dlm's. If I pass 'ptr1', 'insert' and the
> indices for insertion into C I may be able to resize using 'ptr1' realloc,
> shift the data around using pointers and trick the IDL variable structure
> when sending the data back. This sounds risky but at this point all my
> alternatives read, '% Unable to allocate memory: to make array'.
>
> Cheers,
> Randall

--

Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

---------- -------

a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

========== =======