Subject: Re: Big arrays, reducing data
Posted by Jean H. on Wed, 21 Mar 2007 21:26:25 GMT
View Forum Message <> Reply to Message

Eric Hudson wrote:
> Hi,
>
> I have what I hope is an easy question and then probably a hard one.
>
> 1) I need to make some big arrays (ideally 16000^2 elements or more)
> but find I often get "unable to allocate memory" errors. Is there
> some way of determining (at run time) the largest array that I can
> make? In C, for example, I'd try to allocate the memory and check for
> whether it was allocated, then cut the array size if it wasn't. Is
> there an equivalent technique in IDL?

There is the memTest procedure made by ITTVIS that displays the 10
biggest array that you can store. I have modified this procedure so you
can retrieve a) the size of the biggest array you can save and b) the
TOTAL available memory.

Here is a copy of the code... note that I just made small modification
to the header as I did not intend to distribute this code.
The calling sequence is:
biggestArrayInBits = availableMemory()
or biggestArrayInBits = availableMemory(TotalAvailableMemoryInBits)

Hope that helps!
Jean

```
; function: availableMemory   (previously: Procedure: MEMTEST)
;
;
;http://www.rsinc.com/services/techtip.asp?ttid=3441
;
; Syntax: memtest
;
; Purpose:
; This procedure was designed primarily to test the impacts of Windows OS
; memory fragmentation behavior on IDL memory allocation.
;
; The procedure attempts to allocate 10 coexisting memory blocks of 2 GB
size.
; If there is not enough memory to accomplish this, it allocates the 10
; largest coexisting blocks that it can. It stops allocating new memory
blocks
; either:
;
```

```
; - when it has allocated full 10 blocks.
; - when it cannot allocate any additional block of more than 1 MB in size
;   (i.e. when the application has run out of available memory).
;
; Postcondition:
; This program outputs a log of its successful allocations that may look
like:
;
; Memory block # 1: 1168 Mb (total: 1168 Mb)
; Memory block # 2:  206 Mb (total: 1374 Mb)
; Memory block # 3:  143 Mb (total: 1517 Mb)
; Memory block # 4:  118 Mb (total: 1635 Mb)
; Memory block # 5:   79 Mb (total: 1714 Mb)
; Memory block # 6:   54 Mb (total: 1768 Mb)
; Memory block # 7:   41 Mb (total: 1809 Mb)
; Memory block # 8:   39 Mb (total: 1848 Mb)
; Memory block # 9:   31 Mb (total: 1879 Mb)
; Memory block #10:   16 Mb (total: 1895 Mb)
;
; (Note that the output may have fewer than 10 blocks of memory)
;
;
;
;MODIFICATION:
;February 2007: Jean-Gabriel Hasbani, jghasban@ucalgary.ca
;This is now a function that returns the size, in MB, of the biggest
array the memory could hold.
;If specified by the argument, the total availalbe memory can also be
saved.
;The showAll keywords allows one to print all the availble memory that
can be used by the biggest arrays
;
; maxArraySize = availableMemory(totalSize, /showAll)
;-
function availableMemory, totalSize, showAll  = showAll
   compile_opt idl2 ; set default integers to 32-bit and enforce [] for
indexing

 biggestArray = 0ull

   MB = 2^20
   currentBlockSize = MB * 2047   ; 2 GB
   maxIterations = 10             ; Max loop iterations
   memPtrs = ptrarr(maxIterations)
   memBlockSizes = ulonarr(maxIterations)

   for i=0, maxIterations-1 do begin
   ; Error handler
```

```
     catch, err

     ; Sepcifically designed for "Failure to allocate memory..." error
     if (err ne 0) then begin
       currentBlockSize = currentBlockSize - MB     ; ...try 1 MB
smaller allocation
       if (currentBlockSize lt MB) then break  ; Give up, if
currentBlockSize < 1 MB
     endif

   ; This 'wait' enables Ctrl-Break to interrupt if necessary (Windows).
     wait, 0.0001

   ; Allocate memory (if possible)
     memPtrs[i] = ptr_new(bytarr(currentBlockSize, /nozero), /no_copy)
     memBlockSizes[i] = currentBlockSize   ; Store the latest successful
allocation size
     if i eq 0 then biggestArray = currentBlockSize * 8ull;Bits
;currentBlockSize ;byte  ;ishft(currentBlockSize, -20) ;Mb

 ; Print the current allocated block size and the running total, in Mb
 If keyword_set(showAll) then $
     print, format='(%"Memory block #%2d: %4d Mb (total: %4d Mb)")', $
        i + 1, ishft(currentBlockSize, -20),
ishft(ulong(total(memBlockSizes)), -20)
   endfor



   ptr_free,memPtrs

    totalSize = ulong64(total(memBlockSizes)* 8ull) ;bits
;ishft(ulong(total(memBlockSizes)), -20) ;Mb

 return, biggestArray
end
```