Subject: Memory Headache II
Posted by David on Fri, 30 Jan 2004 21:06:43 GMT
View Forum Message <> Reply to Message

Well, I have a lovely little Mac with 8 GB of RAM, and an allegedly
64-bit OS. I have learned that OS X is not capable of giving more than
2^32 bytes of address space to a single process. I find IDL gives up on
data memory around 3.6 GB. Does IDL hold a chunk of RAM in reserve for
compiled code and temporary variables for operations?

IDL's 32 bit implementation is supposed to be capable of 2 GB arrays.
When I attempt to grab more than 2 GB using the new_ptr function I get
the expected malloc errors. However, I find I get these errors even if
I try to grab something like 1.6 or 1.7 GB; a value too far off to be
attributable to whether a GB is 10^9 bytes or 2^30 bytes. Is there some
unseen overhead at issue here? (I have experimented in detail to find
out to the byte how far I can go, but do not have that info handy.)

Another oddity occurs when I try to

a=fltarr(1024,1024,1024,/noz)

Instead of the stream of malloc errors, I get something to the effect
of "this array has too many elements" . Is there an element limit
too? When I try

a=bytarr(1024,1024,1024,/noz) the memory is allocated w/o a hitch.

My bottom line questions:
1) Why can't I get 2 GB arrays?
2) What is this "too many elements thing?" Does idl really care about
the number of elements or is this some sort of memory error anticpator
that kicks in under certain circumstances to avoid even calling malloc
with too large of a request?
3) Does anyone have a finger in the wind as to when a full 64-bit
implementation of IDL might be available for *nix distributions?

Thanks for your input

--
David Theil davidt123 ####at####sbcglobal####dot####net