Subject: Re: dymamic memory allocation
Posted by Mark Hadfield on Mon, 06 Dec 2004 20:50:03 GMT
View Forum Message <> Reply to Message

Marc Reinig wrote:
> "Mark Hadfield" <m.hadfield@niwa.co.nz> wrote in message
> news:cp0fej$j1r$1@newsreader.mailgate.org...
>
>> ...(The array created in line 1 is no
>> longer accessible to IDL and the memory associated with it may--or may
>> not--be returned to the operating system.)
>
> When would it be returned to the OS? Clearly (I hope) when IDL was shut
> down. How about when a break occurs? It would seem that a program could
> inadvertantly eat up much of the available memory this way.
>
> Does IDL have a background process that eventually free's memory that is no
> longer associated with variables?
>
> How would I manually free the memory or tell IDL to?

The details of IDL's memory management depend on the platform. There have been threads about this over the years on this newsgroup. As I understand it, IDL allocates and frees memory via calls to the C functions "malloc" and "free". What these functions do is up to the run-time library. I believe that on some platforms (in the past, if not now) memory released by "free" was not actually available to other processes until IDL exited.

I can report that on Windows 20000, allocation and freeing of memory seems to occur quickly. Eg, if I enter the following at a command prompt:

IDL> a = fltarr(100000000)

the VM Size for "idlde.exe" reported by Windows Task Manager goes from 10,164 kiB to 401,176 kiB within a second or two. If I then type

IDL> a = 0

it goes back to 10,164 kiB.

In the case of ordinary variables, there is no need for a "background process" to manage memory. It is straightorward to determine which blocks of data are being used and which are not, and this is presumably done by the IDL interpreter after every statement is executed (or perhaps somewhat less often). Heap variables (pointers and objects) are more difficult as there can be more than one reference to each variable. IDL *could* implement an automatic "garbage collector" that frequently

monitors the status of the heap variables to determine when they can be discarded. (Many other languages--eg Java, Python, Matlab--do this.) But IDL leaves this for the programmer to do, with the tools OBJ_DESTROY, PTR_FREE, HEAP_GC and HEAP_FREE.

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)