Subject: Re: How to expand large arrays?
Posted by rivers on Wed, 19 Apr 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article <3n142g$j4k@aplinfo.jhuapl.edu>, art.croucher@jhuapl.edu writes:
> I am trying to read in a large but unknown amount of data, and am
> getting (IDL 3.6.1 VMS and Windows) 'unable to allocate core' crashes
> long before I should be out of memory. Is there something that causes
> IDL to allocate something like 5x the size of the array being used? I
> understand that a temporary array is created, but I thought this would
> limit me to an array 1/2 the size of the available memory.
> I create a large array and expand it by either concatenation or by
> creation of a new array and pasting. In both instances, I crash when
> using 1/5 of my memory quota and less than half the physical memory
> available (according to IDL help,/memory. The respective operating
> systems say I'm at the limit of both physical and virtual). Yes, I
> created enough huge images to use my expected quota - got just the
> numbers I expected.
> Can anybody tell me what is going on, and how can I create an array
> that is large but nowhere near the quota?
> The two methods I tried were:
> temp=(~10000,2)
> data=[temporary(data),temp]
> and
> temp=fltarr(npts+10000,2)
> temp(0,0)=data
> data=temp
> temp=0
> Both crashed when the IDL-reported memory usage was 9MB of my 50 MB VMS
> quota (VMS said I was using all 50MB).

I think this is discussed in the FAQ, (and if not it should be!). The problem
is memory fragmentation. When you do:
IDL> data=[temporary(data),temp]
you release the memory which IDL was using for "data". However, IDL now needs
to allocate a larger CONTIGUOUS chunk of memory for the new larger version of
"data". It can't re-use any of the memory you just released for this purpose,
because it won't be part of a contiguous free memory region. The problem just
continues as your array grows bigger and bigger, all of the previously freed
memory is unusable for your growing array. It will be re-used whenever possible
for future smaller arrays. Each time IDL calls "malloc" the operating system
gives it more memory, (using up your VIRTUALPAGCNT), but that virtual memory
remains allocated to your process (calls to "free" do not return the memory to
VMS). The same behavior is found in UNIX systems.

The solution to your problem is to allocate memory for your growing array in
fewer, larger chunks. Ideally, if you have an upper bounds on how big the

array could possibly get, allocate the memory all at once, i.e.
IDL> data = fltarr(1e6, /nozero)
The /nozero flag speeds things up if you are going to overwrite all of the memory anyway.

It would be nice if there were a memory compaction routine, but I think this is really a problem with the C runtime library and not with IDL itself.

Mark Rivers	(312) 702-2279 (office)
CARS	(312) 702-9951 (secretary)
Univ. of Chicago	(312) 702-5454 (FAX)
5640 S. Ellis Ave.	(708) 922-0499 (home)
Chicago, IL 60637	rivers@cars3.uchicago.edu (Internet)