Subject: Re: IDL Memory Management
Posted by Craig Markwardt on Thu, 06 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

"Myron Brown" <Myron.Brown@jhuapl.edu> writes:

> Hi. I'm having a problem that I'm not sure how to solve. I call a
> procedure I wrote which manipulates large images. When it's done, it passes
> me back a relatively small array. However, AFTER the procedure is complete,
> I have problems with swap space. It appears that something is still
> allocated (or something). I am not using pointers, just simple arrays
> (which I assume get thrown on the stack - maybe not). Any ideas on how to
> deallocate this?
>

This question comes up every so often, and there is no easy answer.
RSI has a "Tip" on its web page which treats this issue in a little
more detail.

I assume you are running under a Unix-type OS. Under Unix the memory
space of any process including IDL is treated as a one-demensional
array of cells. IDL requests more memory from the system on an
as-needed basis. If you do a lot of creating and destroying of IDL
variables, especially with big variables, you can end up with a lot of
unused *HOLES* in memory, and unfortunately holey memory can't be
returned to the system. Windows may or may not be different in this
respect.

The solutions are to:

* avoid making extra copies of your memory hogging image variables.
  This includes the implicit copies that are made in arithmetic
  expressions.

* read about the TEMPORARY() function, and use it.

* consider some form of "chunking" in your processing. That is,
  operate on banded subsets of your image so that your overall memory
  footprint is smaller. I do this successfully with gigabytes worth
  of data.

Cheers, and good luck,

Craig

--
-- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu

Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: IDL Memory Management
Posted by davidf on Fri, 07 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Hi Folks,

I ignored my own good judgement completely earlier
today and wrote this:

> As I understand it, the compiler used for the Windows versions
> of IDL *can* return process memory back to the OS. But it is
> the only one to do so. It is not a feature, apparently, of
> UNIX, Mac, and VMS compilers, which rely on Malloc and Free
> for dynamic memory allocation.

This turns out to be completely boneheaded, as I've now
been informed by people who really know how this works.

Windows is no different from UNIX or Mac or any other
operating system. Dynamic memory allocation is pretty
much the same on all of them -- malloc. And there is
just no "general" was to do the memory allocation and
give unused memory back to the OS.

Sorry to have gotten your hopes up. The good news is that
you can sell all that Microsoft stock you just bought at
a nice premium, probably. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: IDL Memory Management
Posted by Craig Markwardt on Fri, 07 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:

> Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:
>
>> Windows used to have a non-linear addressing model when physical
>> system memory was scarce. Memory was allocated in chunks which could
>> be moved around by the Windows as needed, thus alleviating the above
>> fragmentation problem. I think however that more recent versions of
>> Windows have accepted the linear "Unix" model of memory architecture
>> that I have described.
>
> As I understand it, the compiler used for the Windows versions
> of IDL *can* return process memory back to the OS. But it is
> the only one to do so. It is not a feature, apparently, of
> UNIX, Mac, and VMS compilers, which rely on Malloc and Free
> for dynamic memory allocation.

I suppose I am picking nits now. The fragmentation problem actually
has nothing to do with the Unix compilers, malloc or free. The
problem is that Unix-style processes have memory that is one
dimensional without holes. [To get technical, blame it on the
internal brk() function.]

Windows NT also has a single linear memory space for processes, but it
may possibly have more features to deal with fragmentation
efficiently.

Nit-ingly yours,
Craig

--
-- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: IDL Memory Management
Posted by davidf on Fri, 07 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:

> Windows used to have a non-linear addressing model when physical
> system memory was scarce. Memory was allocated in chunks which could
> be moved around by the Windows as needed, thus alleviating the above
> fragmentation problem. I think however that more recent versions of
> Windows have accepted the linear "Unix" model of memory architecture

> that I have described.

As I understand it, the compiler used for the Windows versions
of IDL *can* return process memory back to the OS. But it is
the only one to do so. It is not a feature, apparently, of
UNIX, Mac, and VMS compilers, which rely on Malloc and Free
for dynamic memory allocation.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: IDL Memory Management
Posted by Craig Markwardt on Fri, 07 Jan 2000 08:00:00 GMT

View Forum Message <> Reply to Message

wbiagiot@suffolk.lib.ny.us writes:
> In article <onso0brt0c.fsf@cow.physics.wisc.edu>,
> craigmnet@cow.physics.wisc.edu wrote:
>>
> If you do a lot of creating and destroying of IDL
>> variables, especially with big variables, you can end up with a lot of
>> unused *HOLES* in memory, and unfortunately holey memory can't be
>> returned to the system. Windows may or may not be different in this
>> respect.
>>
>
> Craig,
>
> Out of curiosity, are you saying that these memory "holes" are never
> reused by UNIX or IDL? (i.e. every time you enter an IDL subroutine,
> create an array, leave the subroutine (destroying the array), that it
> acts like a memory leak?)
>
> BTW, I am a Windows user.

The holes can be reused by IDL. The point however is that once the
memory has been allocated by IDL, it's very hard to unallocated it.
The original question involved a person who was working on very large
images but only returning a small subimage, and the person complained

that their machine began to swap after running the analysis. I
presume that the operations on the large images consumed a lot of
memory which was never returned to the system.

Here is an example of three commands to IDL which can show the
fragmentation problem. The horizontal bar is meant to represent the
linear memory of the IDL process.

```
a = findgen(5000)   ;; Allocates first available memory
|******************* a ***************************|

b = findgen(300)    ;; Allocates next available memory
|******************* a ***************************|*b*|

a = 0               ;; A large hole is left!
|----------------- Unused ----------------------|*b*|
```

In this case, the memory that "a" occupied can't be returned to the
system because "b" is still being used. IDL can still reuse that
unused block for other new variables, as long as it fits in the space.

In principle IDL could occasionally "compact" the memory space to
remove the holes, but this would probably have a large impact on
performance and also cause a lot of internal programming headaches.

Windows used to have a non-linear addressing model when physical
system memory was scarce. Memory was allocated in chunks which could
be moved around by the Windows as needed, thus alleviating the above
fragmentation problem. I think however that more recent versions of
Windows have accepted the linear "Unix" model of memory architecture
that I have described.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: IDL Memory Management
Posted by wbiagiot on Fri, 07 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

In article <onso0brt0c.fsf@cow.physics.wisc.edu>,
  craigmnet@cow.physics.wisc.edu wrote:

>
If you do a lot of creating and destroying of IDL
> variables, especially with big variables, you can end up with a lot of
> unused *HOLES* in memory, and unfortunately holey memory can't be
> returned to the system. Windows may or may not be different in this
> respect.
>

Craig,

Out of curiosity, are you saying that these memory "holes" are never
reused by UNIX or IDL? (i.e. every time you enter an IDL subroutine,
create an array, leave the subroutine (destroying the array), that it
acts like a memory leak?)

BTW, I am a Windows user.

-Bill B.

--
"They don't think it be like it is, but it do."

Oscar Gamble, NY Yankees

Sent via Deja.com http://www.deja.com/
Before you buy.