Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by Gray on Tue, 04 Jan 2011 22:30:18 GMT
View Forum Message <> Reply to Message

On Jan 4, 5:01 pm, Matt Francis <mattjamesfran...@gmail.com> wrote:
> I have some code I've written that looks clunky and I was wondering if
> there is a more efficient (faster and or using less memory) way to do
> this.
>
> I am using a custom object with a member self.foo which will end up
> being a matrix, built up by appending arrays one at a time as I loop
> over each step of a process. This update code currently looks like
> this:
>
> temp = [*(self.foo),next_array]
> ptr_free,self.foo
> self.free = ptr_new(temp)
>
> This seems to be a bit wastefull in terms of how many times memory is
> allocated and deallocated to get the job done. Something simple like
>
> self.free = ptr_new([*(self.foo),next_array]
>
> causes a memory leak due to the dangling pointer. I don't see how the
> TEMPORARY function can be used here without causing a leak.
>
> Any tips from the pros?

Why mess about with ptr_new and ptr_free? Unnecessary.

temp = [*self.foo,next_array]
*self.foo = temp

Or, the minimalist approach:

*self.foo = [*self.foo,next_array]

Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by Matt Francis on Tue, 04 Jan 2011 22:39:00 GMT
View Forum Message <> Reply to Message

On Jan 5, 9:30 am, Gray <grayliketheco...@gmail.com> wrote:
> On Jan 4, 5:01 pm, Matt Francis <mattjamesfran...@gmail.com> wrote:
>
>
>
>> I have some code I've written that looks clunky and I was wondering if

>> there is a more efficient (faster and or using less memory) way to do
>> this.
>
>> I am using a custom object with a member self.foo which will end up
>> being a matrix, built up by appending arrays one at a time as I loop
>> over each step of a process. This update code currently looks like
>> this:
>
>> temp = [*(self.foo),next_array]
>> ptr_free,self.foo
>> self.free = ptr_new(temp)
>
>> This seems to be a bit wastefull in terms of how many times memory is
>> allocated and deallocated to get the job done. Something simple like
>
>> self.free = ptr_new([*(self.foo),next_array]
>
>> causes a memory leak due to the dangling pointer. I don't see how the
>> TEMPORARY function can be used here without causing a leak.
>
>> Any tips from the pros?
>
> Why mess about with ptr_new and ptr_free? Unnecessary.
>
> temp = [*self.foo,next_array]
> *self.foo = temp
>
> Or, the minimalist approach:
>
> *self.foo = [*self.foo,next_array]

Hmmm, I didn't expect that would work! Still keep forgetting IDL 'pointers' aren't really pointers and can do funny things (I'm a C++ programmer and this wouldn't work with a 'real' pointer).

Thanks for your help.

Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by natha on Tue, 04 Jan 2011 22:44:49 GMT
View Forum Message <> Reply to Message

Try this:

temp = [TEMPORARY(*self.foo),next_array]
ptr_free,self.foo
self.free = ptr_new(temp,/NO_COPY)

When you are creating the new pointer you are duplicating memory.
The same occurs when you are retrieving the content of *self.foo. Use
the function TEMPORARY and the keyword NO_COPY, your code will be more
efficient.

Cheers,
nata

Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by Matt Francis on Tue, 04 Jan 2011 22:54:33 GMT
View Forum Message <> Reply to Message

On Jan 5, 9:44 am, nata <bernat.puigdomen...@gmail.com> wrote:
> Try this:
>
> temp = [TEMPORARY(*self.foo),next_array]
> ptr_free,self.foo
> self.free = ptr_new(temp,/NO_COPY)
>
> When you are creating the new pointer you are duplicating memory.
> The same occurs when you are retrieving the content of *self.foo. Use
> the function TEMPORARY and the keyword NO_COPY, your code will be more
> efficient.
>
> Cheers,
> nata

Thanks Nata. I see from the TEMPORARY docs that it works by using some
scratch space IDL keeps allocated on hand. Do you know if there is an
upper limit on the size of the arrays you use this approach with
before you don't gain any efficiency (or start to be slower than not
using TEMPORARY), for instance because you are using more memory than
IDL keeps on hand for use with TEMPORARY?

Subject: Re: More efficient method of appending to arrays when using pointers?
Posted by natha on Wed, 05 Jan 2011 03:07:07 GMT
View Forum Message <> Reply to Message

I don't know but I don't thing so. The use of the TEMPORARY function
allows to save time programming with large amounts of data (in my
experience) I didn't find any contradictions to this.
I think that the use of TEMPORARY is always better. You will lose
efficiency using small data (for example, single scalars or small
arrays).

The simple reason is that you are not duplicating memory. In the case discussed above, you are not copying the content of your pointer, you are just retrieving it.

You can do your own tests, for example:

```
  a=PTR_NEW(BYTARR(10000000),/NO_COPY)

  tt=SYSTIME(/SEC)
  b=*a
  PRINT, SYSTIME(/SEC)-tt

  tt=SYSTIME(/SEC)
  b=TEMPORARY(*a) ;; do not forget that you are losing the content of
your pointer
  PRINT, SYSTIME(/SEC)-tt
```

nata
