Subject: Re: Merits of different ways of 'extending' arrays
Posted by Andy Sayer on Tue, 17 Sep 2013 17:40:35 GMT
View Forum Message <> Reply to Message

Right, but for this particular piece of code, I do know the maximum possible size (it's a standalone function I will call for one piece of analysis, as opposed to something I am plugging into the guys of a larger project), so it's good for this application. :)

On Monday, September 16, 2013 10:47:38 AM UTC-4, suicida...@gmail.com wrote:
> The only problem with that is, what is "big enough"? It's going to change from application to application. What happens when your assumption of "big enough" breaks down? Do you have support for re-allocating the arrays when you hit their limits? In order to avoid this you have to make the array SO big that you can start to run into significant memory allocation delays, even when loading just a small amount of data.
>
>
>
> Allocating and expanding in fixed "blocks" as suggested before is a way to elegantly handle this problem, however the block size needs to be tuned for every application or you can start to get some big slowdowns.
>
>
>
> Just some things to consider when choosing your approach.
>
>
>
> On Saturday, September 14, 2013 7:11:07 PM UTC-6, AMS wrote:
>
>> Thanks for the continuing tips!
>
>>
>
>>
>
>>
>
>> The first suggestion (allocate a 'big enough' array up-front, rather than continually extend) worked great for my purposes, so that's what I stuck with, given that it was also very simple. Although I appreciate the continued suggestions.
>
>>
>
>>
>
>>
>
>> Andy

>
>>
>
>>
>
>>
>
>> On Tuesday, September 10, 2013 4:08:40 PM UTC-4, suicida...@gmail.com wrote:
>
>>
>
>>> On Monday, September 9, 2013 4:41:10 PM UTC-6, suicida...@gmail.com wrote:
>
>>
>
>>>
>
>>
>
>>>> Another option is to set up a pointer array nfiles long before the loop, inside the loop load the file and find the valid points, then put that array into that file's pointer, while incrementing a counter to keep track of the total number of points. When you're done, you have all of your data saved in pointers (one per file), and a count of the total number of valid points. Then you allocate your array, loop back through the elements of the pointer array, and fill the array as necessary. Something like:
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>  f = file_search(path, count=nfiles)
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  ptrs = ptrarr(nfiles)
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  num = 0l
>
>>
>
>>>
>
>>
>
```

>>>>
>
>>
>
>>>
>
>>
>
>>>> for i=0l,nfiles-1 do begin
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>   ;; load contents of file
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>   is_valid = where(stuff, n_valid)
>
>>
>
>>>
>
>>
>

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>    if n_valid gt 0 then begin
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>      num += n_valid
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>      ptrs[i] = ptr_new(f.var_1[is_valid])
>
>>
>
>>>
>
>>
>
```

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>    endif
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  endfor
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
```

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>  data = fltarr(num)
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  idx = 0l
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  for i=0l,nfiles-1 do begin
>
>>
>
>>>
>
>>
>
```

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>    if ptr_valid(ptrs[i]) then begin
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>       num = n_elements(*ptrs[i])
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>       data[idx:idx+num-1] = *ptrs[i]
>
>>
>
>>>
>
>>
>
```

```
>>>>
>
>>
>
>>>
>
>>
>
>>>>       ptr_free, ptrs[i]
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>     endif
>
>>
>
>>>
>
>>
>
>>>>
>
>>
>
>>>
>
>>
>
>>>>  endfor
>
>>
>
>>>
>
>>
>
```

>>>
>
>>
>
>>>
>
>>
>
>>> Wish I could edit my post...
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> There should be an "idx += num" next to the ptr_free at the end of the second loop.
