Subject: Array concatenation
Posted by maye on Wed, 07 Jun 2006 10:41:10 GMT
View Forum Message <> Reply to Message

Hi folks,
I was searching for an effective way to do this, but can't find anybody writing about what (as usual) seems to be an obvious problem/task to me. :)
I don't know how many elements I will collect while scanning a bunch of images, so I want to use array concatenation to collect the values like this:
means = [means, currMean]
But for this to compile/run properly, I need mean to exist before.
If I do a
means = 0.
before the loop, I will always have a zero-element in the beginning that I don't want at plotting time. Of course I could remove it with
means = means[1:*]
but not only does this waste resources, it also becomes cumbersome if I collect 20 different data values, for that I ALL have to remove the first value?
Surely there must be a better way?
Please help me to program IDL efficiently! :)
Best regards,
Michael

Subject: Re: Array concatenation
Posted by Peter Clinch on Tue, 17 Jul 2007 07:24:31 GMT
View Forum Message <> Reply to Message

Julio wrote:
> Hi there,
>
> I have a problem with array concatenation. Suppose:
>
> A = bytarr(10, 10)
> B = bytar(10, 5)
>
> C=[A,B]
>
> Sure, it doesn't work, because dimensions don't agree. But, I need to
> join A and B. One possibility is to change the dimensions of array B
> and fill missing lines with zeros. Is it possible??

Not so elegant, but workable, is:

c=bytarr(10,15)

c[*,0:9] = a
c[*,10:14] = b

Pete.
--
Peter Clinch Medical Physics IT Officer
Tel 44 1382 660111 ext. 33637 Univ. of Dundee, Ninewells Hospital
Fax 44 1382 640177 Dundee DD1 9SY Scotland UK
net p.j.clinch@dundee.ac.uk http://www.dundee.ac.uk/~pjclinch/

Subject: Re: Array concatenation
Posted by Paolo Grigis on Tue, 17 Jul 2007 07:28:17 GMT
View Forum Message <> Reply to Message

Julio wrote:
> Hi there,
>
> I have a problem with array concatenation. Suppose:
>
> A = bytarr(10, 10)
> B = bytar(10, 5)
>
> C=[A,B]
>
> Sure, it doesn't work, because dimensions don't agree. But, I need to
> join A and B. One possibility is to change the dimensions of array B
> and fill missing lines with zeros. Is it possible??
>
> comments welcome,
> Best,
> Julio
>

Just use C=[[A],[B]]

IDL> help,a,b,c
A BYTE = Array[10, 10]
B BYTE = Array[10, 5]
C BYTE = Array[10, 15]

Ciao,
Paolo

Subject: Re: Array concatenation

Posted by Conor on Tue, 17 Jul 2007 12:11:57 GMT
View Forum Message <> Reply to Message

On Jul 17, 3:28 am, Paolo_Grigis <pgri...@astro.phys.ethz.ch> wrote:
> Julio wrote:
>> Hi there,
>
>> I have a problem with array concatenation. Suppose:
>
>> A = bytarr(10, 10)
>> B = bytar(10, 5)
>
>> C=[A,B]
>
>> Sure, it doesn't work, because dimensions don't agree. But, I need to
>> join A and B. One possibility is to change the dimensions of array B
>> and fill missing lines with zeros. Is it possible??
>
>> comments welcome,
>> Best,
>> Julio
>
> Just use C=[[A],[B]]
>
> IDL> help,a,b,c
> A BYTE = Array[10, 10]
> B BYTE = Array[10, 5]
> C BYTE = Array[10, 15]
>
> Ciao,
> Paolo

I would go with Paolo's suggestion, unless you absolutely have to join
in left right, instead of up down. In that case, you could try
something like:

c = [[B],[bytarr(10,5)]]
D = [A,C]

Subject: Re: Array concatenation
Posted by Julio[1] on Tue, 17 Jul 2007 13:40:33 GMT
View Forum Message <> Reply to Message

Thanks guys... problem solved.
Best!

Subject: Re: Array concatenation
Posted by Vince Hradil on Tue, 17 Jul 2007 16:25:27 GMT
View Forum Message <> Reply to Message

On Jul 17, 2:28 am, Paolo_Grigis <pgri...@astro.phys.ethz.ch> wrote:
> Julio wrote:
>> Hi there,
>
>> I have a problem with array concatenation. Suppose:
>
>> A = bytarr(10, 10)
>> B = bytar(10, 5)
>
>> C=[A,B]
>
>> Sure, it doesn't work, because dimensions don't agree. But, I need to
>> join A and B. One possibility is to change the dimensions of array B
>> and fill missing lines with zeros. Is it possible??
>
>> comments welcome,
>> Best,
>> Julio
>
> Just use C=[[A],[B]]
>
> IDL> help,a,b,c
> A BYTE = Array[10, 10]
> B BYTE = Array[10, 5]
> C BYTE = Array[10, 15]
>
> Ciao,
> Paolo

or maybe you want transpose([a,transpose(b)]) ?

Subject: Re: array concatenation
Posted by Allan Whiteford on Wed, 30 Jan 2008 16:21:22 GMT
View Forum Message <> Reply to Message

Ryan. wrote:
> Hi All,
>
> I'm trying to add a 1x2 array to the end of an Nx2 array to obtain a (N
> +1)x2 array using the square bracket concatenation trick, but based on
> my limited understanding of it, I can't get it to work. I've looked
> at JD's tutorial on David's website but I still don't get it.
>

> Here is an example of what I want to do:
> IDL> array = [[1,2,3,4],[1,2,3,4]]
> IDL> print, array
> 1 2 3 4
> 1 2 3 4
>
> I want to add another pair to the array to get this:
> 1 2 3 4 5
> 1 2 3 4 5
>
> I want to execute a command similar to this one but I can't figure out
> the correct number of brackets:
> array = [[array], [[5,5]]]
>
> Thanks,
> Ryan.

Ryan,

```
IDL> array = [[1,2,3,4],[1,2,3,4]]
IDL> array=transpose([[transpose(array)],[[5,5]]])
IDL> print,array
      1       2       3       4       5
      1       2       3       4       5
```

but probably there is a smarter way.

Thanks,

Allan

Subject: Re: array concatenation
Posted by Spon on Wed, 30 Jan 2008 16:38:13 GMT
View Forum Message <> Reply to Message

On Jan 30, 3:56 pm, "Ryan." <rchug...@brutus.uwaterloo.ca> wrote:
> Hi All,
>
> I'm trying to add a 1x2 array to the end of an Nx2 array to obtain a (N
> +1)x2 array using the square bracket concatenation trick, but based on
> my limited understanding of it, I can't get it to work. I've looked
> at JD's tutorial on David's website but I still don't get it.
>
> Here is an example of what I want to do:
> IDL> array = [[1,2,3,4],[1,2,3,4]]
> IDL> print, array
> 1 2 3 4

>   1   2   3   4
>
> I want to add another pair to the array to get this:
>   1   2   3   4   5
>   1   2   3   4   5
>
> I want to execute a command similar to this one but I can't figure out
> the correct number of brackets:
> array = [[array], [[5,5]]]
>
> Thanks,
> Ryan.

You want:
array = [array, rebin([5,5],1,2)]

It's time to invoke that dimension juggling tutorial again! It's one
of JD's other tutorials on David's site.

http://www.dfanning.com/tips/rebin_magic.html

Thanks,
Chris

Subject: Re: array concatenation
Posted by Wox on Thu, 31 Jan 2008 08:43:09 GMT
View Forum Message <> Reply to Message

On Wed, 30 Jan 2008 07:56:51 -0800 (PST), "Ryan."
<rchughes@brutus.uwaterloo.ca> wrote:

> Hi All,
>
> I'm trying to add a 1x2 array to the end of an Nx2 array to obtain a (N
> +1)x2 array using the square bracket concatenation trick, but based on
> my limited understanding of it, I can't get it to work. I've looked
> at JD's tutorial on David's website but I still don't get it.
>
> Here is an example of what I want to do:
> IDL> array = [[1,2,3,4],[1,2,3,4]]
> IDL> print, array
>   1   2   3   4
>   1   2   3   4
>
> I want to add another pair to the array to get this:
>   1   2   3   4   5
>   1   2   3   4   5

>
> I want to execute a command similar to this one but I can't figure out
> the correct number of brackets:
> array = [[array], [[5,5]]]
>
> Thanks,
> Ryan.

Yet another possibility:

```
IDL> array = [[1,2,3,4],[1,2,3,4]]
IDL> array = [array,[[5],[5]]]
IDL> print,array
      1       2       3       4       5
      1       2       3       4       5
```

Subject: Re: array concatenation
Posted by lecacheux.alain on Fri, 03 Oct 2008 11:19:13 GMT
View Forum Message <> Reply to Message

On 3 oct, 13:12, lecacheux.al...@wanadoo.fr wrote:
I got a strange error when using the array concatenation construct
within a large loop (a few thousands).
something like:

```
  b = 0B
  for i=0,999 do begin
    ... compute a = array of bytes (a few 100) ...
    b = [b, a]
    endfor
```

This could crash IDL and even, randomly, crash the system (Win2K).
Nothing found by catching errors.
Is there some limits in the implicit addressing of such arrays ?
The error disappeared when compiling with COMPILE_OPT IDL2.

Subject: Re: array concatenation
Posted by Joost Aan de Brugh on Fri, 03 Oct 2008 15:15:12 GMT
View Forum Message <> Reply to Message

On Oct 3, 1:19 pm, lecacheux.al...@wanadoo.fr wrote:
> On 3 oct, 13:12, lecacheux.al...@wanadoo.fr wrote:
> I got a strange error when using the array concatenation construct
> within a large loop (a few thousands).
> something like:
> b = 0B
> for i=0,999 do begin

> ... compute a = array of bytes (a few 100) ...
> b = [b, a]
> endfor
> This could crash IDL and even, randomly, crash the system (Win2K).
> Nothing found by catching errors.
> Is there some limits in the implicit addressing of such arrays ?
> The error disappeared when compiling with COMPILE_OPT IDL2.

Hello,

Maybe it has something to do with the array descriptor. Anyway, in a large group concatenation is not the most elegant way. In Matlab (a language similar to IDL), you get a warning if you use such a construction. It has to do with the fact that if your array grows, you ask your system for more space. A safer way is to ask for enough space at once.

afh = a few 100

```
b = BytArr(afh*1000) ; Here is where you ask for a lot of space.
for i=0,999 do begin
  ... compute a = array of bytes (a few 100) ...
  b[i*afh:(i+1)*afh-1] = a ; Now b does not grow in the loop
end
```

Or use a 2D array

```
b = BytArr(afh,1000) ; Here, you ask for the space again.
for i=0,999 do begin
  ... compute a = array of bytes (a few 100) ...
  b[*,i] = a ; Now b does not grow in the loop
end
```

Cheers,
Joost

b = Reform(b,afh*1000) ; Or b = Reform(b,N_Elements(b))

It is a bit harder if you have different 'a few 100's for each iteration

---

Subject: Re: array concatenation
Posted by lecacheux.alain on Fri, 03 Oct 2008 21:16:02 GMT
View Forum Message <> Reply to Message

On 3 oct, 17:15, Joost Aan de Brugh <joost...@gmail.com> wrote:
> On Oct 3, 1:19 pm, lecacheux.al...@wanadoo.fr wrote:

>
> Hello,
>
> Maybe it has something to do with the array descriptor. Anyway, in a
> large group concatenation is not the most elegant way. In Matlab (a
> language similar to IDL), you get a warning if you use such a
> construction. It has to do with the fact that if your array grows, you
> ask your system for more space. A safer way is to ask for enough space
> at once.
>
> afh = a few 100
>
> b = BytArr(afh*1000) ; Here is where you ask for a lot of space.
> for i=0,999 do begin
> ... compute a = array of bytes (a few 100) ...
> b[i*afh:(i+1)*afh-1] = a ; Now b does not grow in the loop
> end
>
> Or use a 2D array
>
> b = BytArr(afh,1000) ; Here, you ask for the space again.
> for i=0,999 do begin
> ... compute a = array of bytes (a few 100) ...
> b[*,i] = a ; Now b does not grow in the loop
> end
>
> Cheers,
> Joost
>
> b = Reform(b,afh*1000) ; Or b = Reform(b,N_Elements(b))
>
> It is a bit harder if you have different 'a few 100's for each
> iteration

Thanks for your reply. I agree with you that such a programming style
is far from ideal.
My point is that it likely can produce some not obvious array boundary
error (and subsequent IDL crash),
while largest used array sizes remain far below the maximum authorized
one.
Or I missed something ?
alx.

Subject: Re: array concatenation
Posted by David Fanning on Fri, 03 Oct 2008 21:20:47 GMT
View Forum Message <> Reply to Message

lecacheux.alain@wanadoo.fr writes:

> Thanks for your reply. I agree with you that such a programming style
> is far from ideal.
> My point is that it likely can produce some not obvious array boundary
> error (and subsequent IDL crash),
> while largest used array sizes remain far below the maximum authorized
> one.
> Or I missed something ?

I think you are missing the extent to which memory
fragmentation reduces the "maximum authorized size"
of an array. In other words, programming with memory
fragmentation issues in mind will, in the end, save
you an enormous amount of unexplained grief.

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: array concatenation
Posted by Karl[1] on Sat, 04 Oct 2008 01:11:19 GMT
View Forum Message <> Reply to Message

On Oct 3, 3:16 pm, lecacheux.al...@wanadoo.fr wrote:
> On 3 oct, 17:15, Joost Aan de Brugh <joost...@gmail.com> wrote:
>
>
>
>> On Oct 3, 1:19 pm, lecacheux.al...@wanadoo.fr wrote:
>
>> Hello,
>
>> Maybe it has something to do with the array descriptor. Anyway, in a
>> large group concatenation is not the most elegant way. In Matlab (a
>> language similar to IDL), you get a warning if you use such a
>> construction. It has to do with the fact that if your array grows, you
>> ask your system for more space. A safer way is to ask for enough space
>> at once.
>
>> afh = a few 100
>

>> b = BytArr(afh*1000) ; Here is where you ask for a lot of space.
>> for i=0,999 do begin
>> ... compute a = array of bytes (a few 100) ...
>> b[i*afh:(i+1)*afh-1] = a ; Now b does not grow in the loop
>> end
>
>> Or use a 2D array
>
>> b = BytArr(afh,1000) ; Here, you ask for the space again.
>> for i=0,999 do begin
>> ... compute a = array of bytes (a few 100) ...
>> b[*,i] = a ; Now b does not grow in the loop
>> end
>
>> Cheers,
>> Joost
>
>> b = Reform(b,afh*1000) ; Or b = Reform(b,N_Elements(b))
>
>> It is a bit harder if you have different 'a few 100's for each
>> iteration
>
> Thanks for your reply. I agree with you that such a programming style
> is far from ideal.
> My point is that it likely can produce some not obvious array boundary
> error (and subsequent IDL crash),
> while largest used array sizes remain far below the maximum authorized
> one.
> Or I missed something ?
> alx.

I dunno, I think that there's something else going on. IDL should fail gracefully if it runs out of memory allocating that array over and over, even if you are chewing up space and causing a lot of fragmentation. How big is the array actually getting when you crash? I just tried it on linux with an array size of 500 and looping 25,000 times with no issue at all. Are you pushing up against the max virt mem in your machine? If you are indeed paging, Windows can get a little unstable under excessive paging.

The COMPILE_OPT IDL2 note is interesting too. Recall that if you don't specify this, integers are 16-bit, and 32-bit if you do. What does your code specified by "... compute a = array of bytes (a few 100) ... " do? Does it call a custom DLM? Is there something that would break or overflow if ints are 16-bit? Still, IDL checks array accesses at run time, so a bad array index shouldn't cause a crash. It may be worth taking a closer look at why IDL2 makes a difference.
