Subject: Re: Reading columns of binary data
Posted by Foldy Lajos on Mon, 07 Aug 2006 17:30:01 GMT
View Forum Message <> Reply to Message

On Mon, 7 Aug 2006, Wayne Landsman wrote:

> This is probably more of a feature request than a question, though
> there is a chance the desired feature already exists within IDL.
>
> A FITS binary table might plausibly consist of 500 columns and 500,000
> rows of data in a fixed length binary format.   To read the 32nd
> column there are 2 options:
>
>   (1) Loop over the 500,000 rows, extracting the scalar value for
> the 32nd column for each row, and construct the 500,000 element ouput
> array
>   (2) Read the entire 500,000 x 500 file into memory, and extract
> the 32nd column
>
> (In practice, one probably would use a hybird method of looping over an
> intermediate size buffer.   Also note that an identical problem occurs
> when extracting every nth pixel from an extremely large image on disk.)
>
> I understand that the extraction of a column will never be as fast as
> reading a row of data, because the bytes to be read are not contiguous.
>   But I am hoping that the heavy work can be done at a lower level than
> the IDL syntax.
>
> Erin Sheldon has recently written a C routine BINARY_READ linked to
> IDL via a DLM to efficiently read a binary column (
> http://cheops1.uchicago.edu/idlhelp/sdssidl/umich_idl.html#C CODE).
> (He also has routines ASCII_READ and ASCII_WRITE to do this for the
> less urgent problem of ASCII columns.)   While I might adopt this
> routine, it would be nice for portability reasons if a DLM were not
> necessary.   Say, a new keyword SKIP to READU
>
> IDL> a = fltarr(200)
> IDL> readu, 1, a, skip = 100
>
> to indicate to skip 100 bytes before reading consecutive elements.
>
> It appears that MATLAB already has a function FREAD to support reading
> columns of data.
>
> --Wayne
>
>

Hi,

ASSOC?

```
; openr, 1, ...
arr=assoc(1, data)
result=arr[0:*:100]
```

however, it has a big disadvantage of using huge memory for data. My
feature request would be an "assoc, unit, type, dimensions, [offset]"
system function, as the value of data in never used.

regards,
lajos

Subject: Re: Reading columns of binary data
Posted by news.verizon.net on Mon, 07 Aug 2006 17:41:07 GMT
View Forum Message <> Reply to Message

```
>
>  ; openr, 1, ...
>  arr=assoc(1, data)
>  result=arr[0:*:100]
>
>  however, it has a big disadvantage of using huge memory for data. My
>  feature request would be an "assoc, unit, type, dimensions, [offset]"
>  system function, as the value of data in never used.
>
```

Yes, I agree with that feature request.   But even if it were
implemented I'm not sure that it solves my original question.   From
the documentation for ASSOC about multiple subscripts

"Although the ability to directly refer to array elements within an
associated file can be convenient, it can also be very slow because
every access to an array element causes the entire array to be
transferred to or from memory."

So my impression is that this is not any better than reading the entire
array and subscripting a column.

--Wayne

P.S. Anybody know if the V6.3 documentation below from ASSOC() is still
valid?   I stopped worrying about the block size of file system
years ago.

"Arrays are accessed most efficiently if their length is an integer
multiple of the block size of the filesystem holding the file. Common
values are powers of 2, such as 512, 2K (2048), 4K (4096), or 8K (8192)
bytes. For example, on a disk with 512-byte blocks, one benchmark
program required approximately one-eighth of the time required to read
a 512 x 512-byte image that started and ended on a block boundary, as
compared to a similar program that read an image that was not stored on
even block boundaries. "

Subject: Re: Reading columns of binary data
Posted by Foldy Lajos on Mon, 07 Aug 2006 18:07:10 GMT
View Forum Message <> Reply to Message

On Mon, 7 Aug 2006, Wayne Landsman wrote:

```
>>
>>  ; openr, 1, ...
>>  arr=assoc(1, data)
>>  result=arr[0:*:100]
>>
>>  however, it has a big disadvantage of using huge memory for data. My
>>  feature request would be an "assoc, unit, type, dimensions, [offset]"
>>  system function, as the value of data in never used.
>>
>
>  Yes, I agree with that feature request.      But even if it were
>  implemented I'm not sure that it solves my original question.   From
>  the documentation for ASSOC about multiple subscripts
>
>  "Although the ability to directly refer to array elements within an
>  associated file can be convenient, it can also be very slow because
>  every access to an array element causes the entire array to be
>  transferred to or from memory."
>
>  So my impression is that this is not any better than reading the entire
>  array and subscripting a column.
>
```

I have tried it, and it is not even allowed:

```
result=arr[0:*:100]
% Range illegal for record number in assoc var ref: ARR
```

so, may be assoc is associated with old VAX/VMS assoc in my head :-)

regards,

lajos

Subject: Re: Reading columns of binary data
Posted by JD Smith on Mon, 07 Aug 2006 21:54:44 GMT
View Forum Message <> Reply to Message

On Mon, 07 Aug 2006 10:12:16 -0700, Wayne Landsman wrote:

> This is probably more of a feature request than a question, though
> there is a chance the desired feature already exists within IDL.
>
> A FITS binary table might plausibly consist of 500 columns and 500,000
> rows of data in a fixed length binary format.   To read the 32nd
> column there are 2 options:
>
>   (1) Loop over the 500,000 rows, extracting the scalar value for
> the 32nd column for each row, and construct the 500,000 element ouput
> array
>   (2) Read the entire 500,000 x 500 file into memory, and extract
> the 32nd column
>
> (In practice, one probably would use a hybird method of looping over an
> intermediate size buffer.   Also note that an identical problem occurs
> when extracting every nth pixel from an extremely large image on disk.)
>
> I understand that the extraction of a column will never be as fast as
> reading a row of data, because the bytes to be read are not contiguous.
>   But I am hoping that the heavy work can be done at a lower level than
> the IDL syntax.
>
> Erin Sheldon has recently written a C routine BINARY_READ linked to
> IDL via a DLM to efficiently read a binary column (
> http://cheops1.uchicago.edu/idlhelp/sdssidl/umich_idl.html#C CODE).
> (He also has routines ASCII_READ and ASCII_WRITE to do this for the
> less urgent problem of ASCII columns.)   While I might adopt this
> routine, it would be nice for portability reasons if a DLM were not
> necessary.   Say, a new keyword SKIP to READU
>
> IDL> a = fltarr(200)
> IDL> readu, 1, a, skip = 100
>
> to indicate to skip 100 bytes before reading consecutive elements.
>
> It appears that MATLAB already has a function FREAD to support reading
> columns of data.

But does it work in a vector fashion? Using POINT_LUN with READU in

IDL gives you the FREAD functionality, but requires you to loop
500,000 times (in your example). So, the only real problem you have
is the looping penalty (and it's a big problem).

I'd regard this as yet another example of why IDL needs a fast
compiled "side-loop" primitive for generic looping operations which
don't need the full conveniences of the interpreter loop (ability to
hit Control-C to interrupt, etc.). I should be able to say:

```
a=fltarr(500000,/NO_ZERO)
f=0.0
for_noblock i=0L,500000L-1L do begin
  point_lun,un,i*100L
  readu,un,f
  a[i]=f
endfor
```

and not have it be 100x slower than the equivalent in C.

JD

Subject: Re: Reading columns of binary data
Posted by Wayne Landsman on Mon, 07 Aug 2006 22:21:17 GMT
View Forum Message <> Reply to Message

"JD Smith" <jdsmith@as.arizona.edu> wrote in message
news:pan.2006.08.07.21.54.44.519049@as.arizona.edu...
>>
>> It appears that MATLAB already has a function FREAD to support reading
>> columns of data.
>
> But does it work in a vector fashion?

If I read the documentation correctly I think it does:
....
 http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fr ead.html#482088
When skip is specified, fread reads in, at most, a repetition factor number
of values (default is 1), skips the amount of input specified by the skip
argument, reads in another block of values, again skips input, and so on,
until count number of values have been read. If a skip argument is not
specified, the repetition factor is ignored. Use the repetition factor with
the skip argument to extract data in noncontiguous fields from fixed-length
records.
....
A generic looping operation would still be a nice way of treating similar
problems. --Wayne

Subject: Re: Reading columns of binary data
Posted by JD Smith on Mon, 07 Aug 2006 22:44:21 GMT
View Forum Message <> Reply to Message

On Mon, 07 Aug 2006 22:21:17 +0000, Wayne Landsman wrote:

>
> "JD Smith" <jdsmith@as.arizona.edu> wrote in message
> news:pan.2006.08.07.21.54.44.519049@as.arizona.edu...
>>>
>>> It appears that MATLAB already has a function FREAD to support reading
>>> columns of data.
>>
>> But does it work in a vector fashion?
>
> If I read the documentation correctly I think it does:
> ....
> http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fr ead.html#482088
> When skip is specified, fread reads in, at most, a repetition factor number
> of values (default is 1), skips the amount of input specified by the skip
> argument, reads in another block of values, again skips input, and so on,
> until count number of values have been read. If a skip argument is not
> specified, the repetition factor is ignored. Use the repetition factor with
> the skip argument to extract data in noncontiguous fields from fixed-length
> records.
> ....
> A generic looping operation would still be a nice way of treating similar
> problems. --Wayne

Aha. I suspect you'd have a much better chance getting a similar SKIP keyword added to READU than I would have convincing them of the need to minimize the looping overhead. Your suggestion should be easy for them.

JD

Subject: Re: Reading columns of binary data
Posted by Foldy Lajos on Tue, 08 Aug 2006 14:57:41 GMT
View Forum Message <> Reply to Message

On Mon, 7 Aug 2006, JD Smith wrote:

> I'd regard this as yet another example of why IDL needs a fast
> compiled "side-loop" primitive for generic looping operations which
> don't need the full conveniences of the interpreter loop (ability to
> hit Control-C to interrupt, etc.). I should be able to say:
>
> a=fltarr(500000,/NO_ZERO)

```
>  f=0.0
>  for_noblock i=0L,500000L-1L do begin
>    point_lun,un,i*100L
>    readu,un,f
>    a[i]=f
>  endfor
>
```
> and not have it be 100x slower than the equivalent in C.
>
> JD
>

Hi,

I mentioned assoc yesterday, so let's try again:

```
arr=assoc(un, [0.])
for i=0L,500000L-1L do a[i]=arr[25l*i]
```

it will do the point_lun/readu in one step, so it should be a little bit faster. But I think a faster loop here would not help much, as disk I/O is the limiting factor.

regards,
lajos