On Tue, 25 May 2004 23:03:57 +0000, Jonathan Greenberg wrote:

> I was hoping to get some help/feedback on the use of ASSOC. I'm working
> with a large image file and, while I can extract subsections of the image
> (say 3 consequtive lines), I'd prefer to work with the image as an array,
> but not have it loaded into memory. Is this what ASSOC allows me to do?
> I'm having a problem with the following situation:
>
> Nl and ns = the dimensions of the byte image "someimage.dat".
> Openr,unit,'someimage.dat',/get_lun
> Testassoc=assoc(unit,bytarr(nl,ns))
>
> I'd like to be able to extract the value from position x,y from testassoc --
> how do I do this? Testassoc[0] seems to be the entire image. But
> testassoc[0,1] gives me an end of file error, as does testassoc[0,1,1]
> (which I would think would give me the value at that position).
>
> Using the /packed keyword doesn't seem to do anything. Help!

ASSOC is designed to work with sequential chunks of data, e.g. a 3D
image array of dimension n1xn2xn3 could be indexed with with
a=ASSOC(unit,bytarr(n1,n2)) & a[0..n3-1]. What is not mentioned in
the manual (which is confusing on this topic), is that this only works
for sequential pieces of data. If you have a large image, and you'd
like to work with 2D subsections of the image, ASSOC will not do what
you want. Let's say an image is 2048x2048, and you'd like to work
with a 256x256 subsection somewhere in the middle. This data is not
written contiguously in memory; instead you have to skip over most of
each row to get to the portion of the next row belonging to the
subimage.

A few ways to deal with this problem are:

1. Re-format your image on disk to consist of a series (in memory) of
   sub-array tiles written out serially. Then ASSOC should work fine
   with the fixed tile size you give it.

2. Use the new shared memory mapping in IDL>5.6 to use your systems
   virtual memory subsystem to access non-contiguous tiles of an
   array. The file will only be read from and written to memory as
   needed. You'll need to keep up with the global indexing yourself.
   Keep in mind that you will make the virtual memory system work hard
   as it skips around in memory finding the pieces you need.

JD

Page 2 of 2 ---- Generated from   comp.lang.idl-pvwave archive