

David Fanning writes:

```
> OK, another little data point. Just before I restore
> the save file where I put my simple object, I check
> to see how many objects I have:
>
>   objects = Obj_Valid()
>
> I have 306 of them. That sounds right. (Big application.)
> Now I restore my simple object and check again:
>
>   Restore, Filename='somename.sav'
>   nowObjects = Obj_Valid()
>
> I have 1509 of them! Whoa! What in the world is going on
> here!?
```

OK, are you ready for this? Are you sitting down?
What you have heard about how easy it is to save
and restore objects may be about to change.

My application has one large data set, which is stored
in an ImageCube object. The data is about 3Mbyte in size.
The StudyObject, which I have been trying all day to save,
contains a reference to the ImageCube data, but not the
data itself. By itself, it is small, maybe 100K or so.

OK, so I save the StudyObject. It obviously has a reference
to the ImageCube object, so IDL (helpfully, I think) saves
the ImageCube object, too. But my application is an object
hierarchy. Everything is more or less connected to everything
else. It's a family tree, for God's sake. We are **all** related.

IDL realizes this, thinks it is being helpful, and saves every
object in sight! And although I can't prove it, I think it
saves two backup copies as well, because the entire save file
tops out at a hefty 10 MBytes.

OK, so then I go to restore the small study object. IDL
restores **all** the objects it saved (even though most of
them are truly orphan object (widgets and what-not that
I used last Friday and which are of absolutely no use
to me now). And it also restores its two backup copies, because
no one can accuse IDL of not being thorough!

So I have 310 objects before the restore, and 1513 after the restore. There is a difference of 1203 objects. But I can recover these!

If I use the RESTORED_OBJECTS keyword on the RESTORE command, I can get the references to the 1203 "extra" objects. (I am adrift in a *sea* of objects!)

What should I do with them? Well, I *could* paw through them, discarding the ones that are obviously bogus, but my client got coffee while I was reading the data, so taking another coffee break now is going to be hard on the bladder. So, I take a tip from the Mac folks and put the darn things into a trash container. (Really just an IDL_CONTAINER object labelled "trash".) Now, if I take the trash out when I exit, all is well with the world and no leaking memory.

Here is the code:

```
; Restore the file.
Print, 'Objects Before: ', Obj_Valid()
Restore, File=filename, /Relaxed_Structure_Assignment, $
  Restored_objects=helperObjects
Print, 'Objects After: ', Obj_Valid()
Print, "Helper Objects:", N_Elements(helperObjects)
self.trash -> Add, helperObjects
self.currentStudy = currentStudy
```

Cheers,

David

P.S. Let's just say tomorrow morning, I'm going back to structures. :-)

--

David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>
