Subject: Re: sorting string arrays - non alphabetic and user defined order
Posted by JD Smith on Mon, 23 Oct 2006 21:03:36 GMT
View Forum Message <> Reply to Message

On Tue, 17 Oct 2006 06:00:03 -0600, David Fanning wrote:

> Greg Michael writes:
>
>> hmm... it seems to me that 'result' is the array you are looking for:
>> it has the elements in the order you want, and they have the indices
>> numbered from zero (i.e. there are no empty elements).
>
> It seems that way to me, too. But I had to work with
> the solution for a time before I understood it. It
> would have taken me a LONG time to think of using
> indices rather than strings to come up with a solution.
> Perhaps the solution was TOO concise. I've expanded on
> it just a little bit in this article:
>
> http://www.dfanning.com/idl_way/strsort.html

This is similar to the standard inflate and compare WHERE_ARRAY method. So it would be somewhat simpler (if a bit slower) just to say:

```
mylist=mylist[sort(where_array(mylist,namelist,/PRESERVE_ORD ER))]
```

see turtle.as.arizona.edu/idl/where_array.pro for the modified version which includes PRESERVE_ORDER.

That said, this exhibits the classic defect of "scale-em-up-and-compare" methods which the REBIN/REFORM stuff enables: it starts to get ugly when your comparison vectors get long, scaling as the product of their lengths, and gobbling up enormous amounts of memory in the process. We discuss in detail the pros and cons of the various methods here:

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

For long vectors, you'll be better off with a sort-based algorithm (e.g. ind_int_SORT, see above):

```
mylist=mylist[sort(ind_int_sort(namelist,mylist))]
```

JD