Subject: Re: Combing structures
Posted by Michael Williams on Mon, 16 Aug 2010 19:00:38 GMT
View Forum Message <> Reply to Message

On 2010-06-07 17:01:42 +0200, Michael Williams said:
> xx = findgen(5)
> y = 1.0
> str1 = {one: xx, two:y}
>
> xx = findgen(3)
> y = -1.0
> str2 = {three: xx, four: y}
>
> I want to merge them into one flat structure of their elements. At the
> moment I am doing this manually:

I now have a related question. If

```
IDL> x = {a: [1,2], b: [3,4]}
IDL> y = {a: [5,6], b: [7:8]}
```

then doing

```
IDL> z = [x, y]
```

works, but yields, for example,

```
IDL> print, z.a
      1       2
      5       6
IDL> help, z.a
<Expression>    INT       = Array[2, 2]
```

I would like to concatenate two structures with matching tags by simply appending arrays, so that I get a structure in which

```
IDL> print, z.a
      1       2       5       6
IDL> help, z.a
<Expression>    INT       = Array[4]
```

Is there a built-in way of doing this? The structures I am concatenating will always be defined such that the individual tags can be concatenated trivially like

```
IDL> a = [x.a, y.a]
IDL> print, a
      1       2       5       6
```

but the only way I can think of doing this for the entire structure is
by iterating over the tags and using the execute procedure
[implementation after the signature], which is obviously spectacularly
inelegant. Is there a better way?

-- Mike

[*] Like this. test_append_struct shows it in use.

```
function append_struct, tx, ty
   tags = tag_names(tx)
   s = 'tmp = create_struct("' + tags[0] + '", [tx.' + tags[0] + $
      ', ty.' + tags[0] + '])'
   void = execute(s)
   for i = 1, n_elements(tags) - 1 do begin
      s = 'tmp = create_struct(tmp, "' + tags[i] + '", [tx.' + tags[i] + $
         ', ty.' + tags[i] + '])'
      void = execute(s)
   endfor
   return, tmp
end

pro test_append_struct
   p1 = {x: 0, y:1, z: 1.5}
   p2 = {x: -1, y:2, z: 4}
   p = append_struct(p1, p2)
   help, p, /struct
   print, p.x
   print, p.y
   print, p.z
   p = append_struct(p, p2)
   help, p, /struct
   print, p.x
   print, p.y
   print, p.z
end
```