Subject: Re: recursive tag_names
Posted by davidf on Sun, 22 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Cathy (csaute3@alumni.umbc.edu) writes:

> Does anyone have a "recursive" tag_names like function. I would like to
> pass in a structure and get back a string array with members down to the
> bottom level. For example:
>
> input = {a:{m,n}, b:{r,s,t}, c, d, e}
>
> output = recursive_tag_names_function(input)
>
> print, output
>
> 'input.a.m', 'input.a.n', 'input.b.r', 'input.b.s', 'input.b.t',
> 'input.c', 'input.d', 'input.e'

It must have been a slow news day, or something, but for
some reason this question intrigued me. I thought something
like this might be useful for the object programming I am
doing.

So here it is. There is a little main-level program below
the two program modules that exercises the principle program,
Get_Tags. If you just pass the structure to the program,
you get all the fields with a "dot" in front of the names.
For example,

```
tags = Get_Tags(struct)
```

returns:

```
.one
.one.two
.one.two.three
```

etc. If you want the full name back, pass a second positional
parameter that is the root name of the structure. For example,

```
tags = Get_Tags(struct, 'struct')
```

returns:

```
struct.one
struct.one.two
struct.one.two.three
```

Note that the string vector that is actually returned
from the Get_Tags function (line Tag_Names) is all UPPERCASE.

Cheers,

David

```
 ;------------------------------------------------------------ ---------
Function All_Tags, structure, rootname

; This is a function that recursively searches through
; a structure tree, finding ALL of the structure's field names.
; It returns a pointer to an array of pointers, each pointing
; to the names of structure fields.

IF N_Elements(rootname) EQ 0 THEN rootname = '.' ELSE $
   rootname = StrUpCase(rootname) + '.'
names = Tag_Names(structure)
retValue = Ptr_New(rootname + names)

   ; If any of the fields are structures, report them too.

FOR j=0,N_Elements(names)-1 DO BEGIN
   ok = Execute('s = Size(structure.' + names[j] + ')')
   IF s[s[0]+1] EQ 8 THEN BEGIN
      newrootname = rootname + names[j]
      theseNames = Call_Function('All_Tags', $
         structure.(j), newrootname)
      retValue = [[retValue],[theseNames]]
   ENDIF
ENDFOR

RETURN, retValue
END
 ;------------------------------------------------------------ --------



FUNCTION Get_Tags, structure, rootname

; This function returns the names of all structure fields
; in the structure as a string array. The names are given
; as valid structure names from the root structure name,
; which can be passed in along with the structure itself.

On_Error, 1
```

```
   ; Check parameters.

CASE N_Params() OF

   0: BEGIN
      Message, 'Structure argument is required.'
      ENDCASE

   1: BEGIN
      rootname = ''
      s = Size(structure)
      IF s[s[0]+1] NE 8 THEN $
         Message, 'Structure argument is required.'
      ENDCASE

   2: BEGIN
      s = Size(structure)
      IF s[s[0]+1] NE 8 THEN $
         Message, 'Structure argument is required.'
      s = Size(rootname)
      IF s[s[0]+1] NE 7 THEN $
         Message, 'Root Name parameter must be a STRING'
      ENDCASE

ENDCASE

tags = All_Tags(structure, rootname)

   ; Extract and free the first pointer.

retval = [*tags[0,0]]
Ptr_Free, tags[0,0]

   ; Extract and free the the rest of the pointers.

s = Size(tags)
FOR j=1,s[2]-1 DO BEGIN
   retval = [retval, *tags[0,j]]
   Ptr_Free, tags[0,j]
ENDFOR
Ptr_Free, tags

   ; Return the structure names.

RETURN, retval
END
```

```
   ; Main-level program to exercise Get_Tags.

d = {dog:'spot', cat:'fuzzy'}
c = {spots:4, animals:d}
b = {fast:c, slow:-1}
a = {cars:b, pipeds:c, others:'man'}
tags = Get_Tags(a)
s = Size(tags)
For j=0,s[1]-1 Do Print, tags[j]
END
```

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/