Subject: Re: determining if a structure has a given tag
Posted by mankoff on Wed, 17 Jun 2009 10:00:43 GMT
View Forum Message <> Reply to Message

On Jun 17, 12:25 am, Jeremy Bailin <astroco...@gmail.com> wrote:
> I'm sure there is a simple answer to this, but I can't find it.
>
> Does someone know of a programmatic way to determine if a given tag is
> defined in a structure, without resorting to TAG_NAMES (which seems
> downright ugly). I can do something like
>
> if n_elements(variable) eq 0
>
> to determine if a variable has been defined, but to figure out if a
> structure tag is defined I need to grab a list of all tags?? I guess I
> want something that's functionally equivalent to:
>
> if total(tag_names(struct) eq strupcase(tag)) gt 0
>
> but which doesn't make me shudder. The closest discussion I can find
> on this is here:http://groups.google.com/group/comp.lang.idl-pvwave/bro wse_thread/thr...
> ...but most of that focuses on the peculiarities of dealing with
> _EXTRA, not the more generic case.
>
> -Jeremy.

I wrote a function a few years ago that can be used on deep / complex
structures to tell if a tag exists. It uses tag_names, but since it is
encapsulated maybe it is more palatable to you. Uses recursion to
check nested structures.

```
;
;+
; NAME:
; XIASTREC
;
; PURPOSE:
;   XIASTREC Is A Structure Tag Recursive Existence Checker
;
; CATEGORY:
; Structures, utility, recursion
;
; CALLING SEQUENCE:
;   result = XIASTREC( struct, tag_list )
;
; INPUTS:
;   Struct: A structure
;
```

```
; OPTIONAL INPUTS:
;   tag_list: A string array that contains the tags, subtags,
;             subsubtags, etc. whose existence you wish to verify
;
; KEYWORD PARAMETERS:
;   DEBUG: Print some debug info
;
; OUTPUTS:
;   This function returns 1 if the struct, or struct.tag, or
;   struct.tag.tag.tag... exists, and 0 if it does not. It returns -1
;   if called incorrectly.
;
; PROCEDURE:
;   1. Check if the struct is valid
;   2. Check if tags were passed in. If so, does tag[0] exist in
struct?
;   3. Recurse, checking if tag[1] exist in the struct that is tag[0].
;
; EXAMPLE:
;   PRINT, XIASTREC( foo ) ; 0, foo not defined
;   PRINT, XIASTREC( !P, 'thick' ) ; 1, !P.THICK exists
;
;   x = { foo:{bar:42}, baz:'Goodbye Cruel World' }
;   PRINT, XIASTREC( x, 'bar' ) ; 1, x.bar exists
;   PRINT, XIASTREC( x, ['foo', 'baz'] ) ; 0, x.foo.baz does not exist
;   PRINT, XIASTREC( x, ['foo', 'bar' ] ) ; 1, x.foo.bar exists
;   PRINT, XIASTREC( x, ['foo', 'bar', '42' ] ) ; 0, bar=42, not bar.
42
;
; MODIFICATION HISTORY:
;  Written by: KDM, 2007-07-15
;
;-

function XIASTREC, s, tags_t, debug=debug,
recursive_level=recursive_level

  if keyword_set(debug) then dbg = 1 else dbg = 0

  if size(s,/type) ne 8 then begin
     if dbg then MESSAGE, "Struct is not of type STRUCT", /CONTINUE
     return, 0
  endif

  ;; only S passed in. It is a struct. We're done.
  if n_params() eq 1 then begin
     if dbg then MESSAGE, "Struct is existent struct", /CONTINUE
     return, 1
```

```
  endif

  tags = STRUPCASE(tags_t)
  ;; if here, two params passed in
  if size( tags, /type ) ne 7 then begin
    MESSAGE, "ERR: Tags must be of type STRING", /CONTINUE
    return, -1
  endif

  names = tag_names( s )
  idx = where( names eq tags[0], count )
  if count eq 0 then begin
    if dbg then MESSAGE, "Tag '"+tags[0]+"' not in struct", /CONTINUE
    return, 0
  endif

  ;; only 1 tag, and it matched
  if n_elements(tags) eq 1 then begin
    if dbg then MESSAGE, "Tag '"+tags[0]+"' found", /CONTINUE
    return, 1
  endif

  if keyword_set(recursive_level) then r = recursive_level else r = 0
  if dbg then begin
    if r eq 0 then MESSAGE, "Beginning recurse...", /CONTINUE

    spaces = ''
    for i=0,r-1 do spaces=spaces+' '
    MESSAGE, spaces + "Recursing to level " + STRTRIM(r,2), /CONTINUE
  endif

  ;; recurse! :)
  exists = XIASTREC( s.(idx), tags[1:*], $
                     debug=debug, recursive_level=r+1 )

  if dbg then MESSAGE, spaces+"Level "+STRTRIM(r,2)+" completed", /
CONTINUE
  return, exists

MESSAGE, "ERR: Shouldn't be here...", /CONTINUE
return, -1
end
```