
Subject: Re: recursive tag_names

Posted by [Richard D. Hunt](#) on Fri, 27 Mar 1998 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

I have written three routines to work with structures. The first is REMOVEELM.PRO which will take a structure and remove an element from it.

The second is REPLACEELM.PRO which will add or replace an element within a structure. The third is STRUCTELMNames.PRO will take a structure

and return a string array which lists every element of the structure.

All of these routines will work on structures of structures. The only real side effect I have seen so far is that if you are using named structures the first two routines will return anonymous structures since you can't change a named structure.

Rich

```
;
;
; This function will remove an element of a structure.
;
;
; NOTE: This will return an anonymous structure since named a
;       named structure cannot be resized once it is defined.
;
;
; str: the structure you want to remove an element from.
; elm: the string name of the element to be removed.
; base: the "." separated string substructure of where to
;       find the element to be removed. If this element has
;       no "." in it then the particular element to be removed
;       will be at the highest level.
;
;
; Example:
;
;
; IDL> str = {a:1, b:{c:2, d:{e:3, f:4}},g:5}
; IDL> nstr = RemoveElm(str, 'f', SubStructure='b.d')
; IDL> print, nstr
; { 1{ 2{ 3}} 5}
;
```

```
FUNCTION RemoveElm, str, elm, SubStructure=base
; Catch any errors.
Catch, error_status
IF error_status NE 0 THEN BEGIN
    print, "Couldn't remove element of structure."
    Return, str
```

```

ENDIF

; Get the names of the elements in str
names = StrLowerCase(Tag_Names(str))
element = StrLowerCase(elm)

IF N_Elements(base) NE 0 THEN BEGIN
    pos = StrLowerCase(StrTrim(Str_Sep(base, '.'), 2))

    FOR i=0, N_Tags(str)-1 DO BEGIN
        type = Size(str.(i))
        IF type[N_Elements(type)-2] EQ 8 AND names[i] EQ pos[0] THEN
BEGIN
            ; Recurse into the structure.
            IF N_Elements(pos) GT 1 THEN BEGIN
                nb = String(Format='(100(A,.;"."))',pos[1:*)
                substr = RemoveElm(str.(i), elm, SubStructure=nb)
            ENDIF ELSE $
                substr = RemoveElm(str.(i), elm)

            ; Add the substructure to the structure.
            IF N_Elements(newstr) EQ 0 THEN $
                newstr = Create_Struct(names[i], substr) $
            ELSE $
                newstr = Create_Struct(newstr, names[i], substr)
            ENDIF ELSE BEGIN
                ; Add the substructure to the structure.
                IF N_Elements(newstr) EQ 0 THEN $
                    newstr = Create_Struct(names[i], str.(i)) $
                ELSE $
                    newstr = Create_Struct(newstr, names[i], str.(i))
                ENDELSE
            ENDFOR
        ENDIF ELSE BEGIN
            FOR i=0, N_Tags(str)-1 DO BEGIN
                IF names[i] NE element THEN BEGIN
                    IF N_Elements(newstr) EQ 0 THEN $
                        newstr = Create_Struct(names[i], str.(i)) $
                    ELSE $
                        newstr = Create_Struct(newstr, names[i], str.(i))
                    ENDIF
                ENDFOR
            ENDELSE

            Return, newstr
        END
    
```

```

;
; This function will add or replace an element of a structure.
;
; NOTE: This will return an anonymous structure since named a
;       named structure cannot be resized once it is defined.
;
; str: the structure you want to remove an element from.
; elm: the string name of the element to be removed.
; val: the value to replace or add to the structure.
; base: the "." seperated string substructure of where to
;       find the element to be removed. If this element has
;       no "." in it then the particular element to be removed
;       will be at the highest level.
;
; Example:
;
;IDL> str = {a:1, b:{c:2, d:{e:3}},g:5}
;IDL> nstr = ReplaceElm(str, 'f', 4, SubStructure='b.d')
;IDL> print, nstr
;{   1{   2{   3   4}}   5}
;

```

FUNCTION ReplaceElm, str, elm, val, SubStructure=base

```

; Catch any errors.
Catch, error_status
IF error_status NE 0 THEN BEGIN
    print, "Couldn't replace element of structure."
    Return, str
ENDIF

; Get the names of the elements in str
names = StrLowerCase(Tag_Names(str))
element = StrLowerCase(elm)

IF N_Elements(base) NE 0 THEN BEGIN
    pos = StrLowerCase(StrTrim(Str_Sep(base, '.'), 2))

    FOR i=0, N_Tags(str)-1 DO BEGIN
        type = Size(str.(i))
        IF type[N_Elements(type)-2] EQ 8 AND names[i] EQ pos[0] THEN
BEGIN
            ; Recurse into the structure.
            IF N_Elements(pos) GT 1 THEN BEGIN
                nb = String(Format='(100(A,.;"."))',pos[1:*])
                substr = ReplaceElm(str.(i), elm, val, SubStructure=nb)
            ENDIF ELSE $

```

```

        substr = ReplaceElm(str.(i), elm, val)

; Add the substructure to the structure.
IF N_Elements(newstr) EQ 0 THEN $
    newstr = Create_Struct(names[i], substr) $
ELSE $
    newstr = Create_Struct(newstr, names[i], substr)
ENDIF ELSE BEGIN
; Add the substructure to the structure.
IF N_Elements(newstr) EQ 0 THEN $
    newstr = Create_Struct(names[i], str.(i)) $
ELSE $
    newstr = Create_Struct(newstr, names[i], str.(i))
ENDELSE
ENDFOR
ENDIF ELSE BEGIN

FOR i=0, N_Tags(str)-1 DO BEGIN
    IF names[i] NE element THEN BEGIN
        IF N_Elements(newstr) EQ 0 THEN $
            newstr = Create_Struct(names[i], str.(i)) $
        ELSE $
            newstr = Create_Struct(newstr, names[i], str.(i))
        ENDIF
    ENDFOR

; Add the new element
    newstr = Create_Struct(newstr, element, val)
ENDELSE

Return, newstr
END

```

```

;
;
; This function will take a structure and return a string array where
; each element represents a particular element in the structure. It
; will recurse substructures as well.
;
; Example:
; IDL> str = {a:1, b:{c:FltArr(2), d:{e:3, f:4}},g:5}
; IDL> print, StructElmNames('str', str)
; str.a str.b.c str.b.d.e str.b.d.f str.g
;
;

```

FUNCTION StructElmNames, sname, struct

; Get a list of all of the elements in the structure.

```
elms = StrLowerCase(Tag_Names(struct))
```

; Build a string array of elements and their values.

```
FOR i=0L, N Tags(struct)-1 DO BEGIN
```

; Figure out what type of element we are dealing with.

```
s = Size(struct.(i))
```

```
type = s[N_Elements(s)-2]
```

```
IF type EQ 8 THEN BEGIN
```

```

; The element was another structure so recurse.

```

```
IF N Elements(outstr) EQ 0 THEN $
```

```
outstr = [StructElmNames(sname+'.'+elms[i], struct.(i)) $
```

ELSE \$

```
outstr = [outstr, StructElmNames(sname+'.'+elms[i],
```

```
struct.(i))]
```

```
ENDIF ELSE BEGIN
```

; The element was something other than a structure.

```
IF N Elements(outstr) EQ 0 THEN $
```

```
outstr=[sname+'.'+elms[i]] $
```

ELSE \$

```
outstr=[outstr,sname+'.'+elms[i]]
```

ENDELSE

ENDFOR

Return, outstr

END

—

Richard D. Hunt

SANDIA NATIONAL LABORATORIES / / /

P.O. Box 5800 M/S 0965 / /

Albuquerque, NM 87185-0965 - / / / / /

Voice: (505)844-3193 / / / /

Fax: (505)844-5993

E-Mail: rdhunt@sandia.gov