Subject: Re: Looping over parameters without EXECUTE()
Posted by Ken Mankoff on Tue, 03 May 2005 23:31:43 GMT
View Forum Message <> Reply to Message

On Tue, 3 May 2005, JD Smith wrote:
> On Tue, 03 May 2005 15:43:33 +0200, Thomas Pfaff wrote:
>> JD Smith schrieb:
>>> On Mon, 02 May 2005 12:10:43 -0400, Wayne Landsman wrote:
>>>> The one case where I haven't figured out how to remove
>>>> EXECUTE() from a program (to allow use with the Virtual
>>>> Machine) is where one wants to loop over supplied parameters.
>>>> For example, to apply the procedure 'myproc' to each supplied
>>>> parameter (which may have different data types) one can use
>>>> EXECUTE() to write each parameter to a temporary variable:
>>
>> How about putting all those parameters into a (named or
>> anonymous) struct? Then you can have different types for each
>> parameter and you're still able to loop over the elements.
>>
>> pro doit, param_struct
>> for i=0, n_tags(param_struct) -1 do begin
>> arg = param_struct.(i) ;->this way you can even store result values
>> myproc, arg
>> param_struct.(i) = arg
>> endfor
>> end
>>
>> Would that be a possibility, or am I missing something?
>
> That works OK, and if you used TEMPORARY() you could cut down the
> data copying penalty. There are two main problems with this
> approach: 1) the user has to create a potentially large struct as
> input and then unpack it as output, which is not convenient from
> the command line, especially for output arguments, and 2) the type
> and size of each argument cannot be changed, thanks to the nature
> of structs.

The user doesn't need to build the structure if you build it for
them. I've taken this approach to allow a small subset of
command-line interactivity with the IDL VM. It allows you to run
procedures that only use keywords. That is, no functions, and no
positional arguments. The only IDL procedure I know of like this is
ERASE [, COLOR=value]

It is quite a hack, and at this point _very_ inefficient (each
keyword copies the struct with e = CREATE_STRUCT(name,value,e). And
I wouldn't deploy it in a large codebase. But it works for small
stuff...

Another major limitation is everything gets turned into a string.
Most of the time this is OK... For example,
  erase, color='128'
works ok... But every once in a while it causes problems.

The code can be called like this:
IDL> .com eextra
IDL> eextra_test
IDLVM> erase, color=128 ; example command with only keyword args
IDLVM> quit
IDL>

And here is the eextra (and helper) functions:

   -k.

--
http://spacebit.dyndns.org/

```
function eextra_build, arr
arr = STRTRIM(arr,2)
e = CREATE_STRUCT("IDLVM_NIL",0B) ; filler
if SIZE( arr, /TNAME ) NE "STRING" then return, e

if SIZE( arr, /N_ELEMENTS ) EQ 1 then begin ; not an array, just 1 string
    if ( STREGEX( arr,"^/", /BOOLEAN ) ) then begin ; /foo
       e = CREATE_STRUCT( STRMID( arr[0], 1 ), 1, e )
    endif else if STREGEX( arr, "=", /BOOLEAN ) then begin ; foo = 42
       nv = STRTRIM(STRSPLIT( arr, "=", /extract ),2)

       ; this line so that x=0 makes x=0b, not x='0' (which is 1)
       if (byte(nv[1]))[0] eq 48 then vv = 0B else vv = nv[1]
       e = CREATE_STRUCT( nv[0], vv, e )
    endif else begin
       ; nothing here. String was 'foo', it is an ARG, not an _EXTRA=e
    endelse
    return, e
endif
extras = stregex(arr,"(.*)[=/](.*)",/extract)
blank = where( extras eq '', nblank, ncomp=ncomp )
if ncomp eq 0 then return, CREATE_STRUCT("IDLVM_NIL",0B)
if nblank ne 0 then extras = extras[where(extras ne '')]

; convert "/foo" to "foo=1"
one = STREGEX(extras,"^/",/BOOLEAN)
```

```
;one = STRCMP(arr,"/",1)
if total(one) ne 0 then begin ; some /keywords were set
    bool = STREGEX(extras,"/(.*)", /extract)
    bool = STRMID(bool,1) + "=1"; + STRING(1)
    extras[where(one eq 1)]=bool[where(one eq 1)]
endif

; name/value pairs from form foo=42;
; nv[ 1, * ] = foo & nv[ 2, * ] = 42
;nv = STREGEX(extras,"(.*)=([0-9a-zA-Z-]*)",/subexpr,/extract)
nv = STRTRIM(STREGEX(extras,"(.+)=(.+)",/subexpr,/extract),2)
n = STRTRIM(reform( nv[ 1, * ] ),2) ; allow spaces i.e. foo =  42
v = STRTRIM(reform( nv[ 2, * ] ),2)
zero = where( v eq '', nz )
if nz gt 0 then v[zero]='0'

;stop
for i = 0, n_elements( n )-1 do begin
; attempt to convert numbers to numbers (right now everything is strings)
    if (byte(v[i]))[0] eq 48 then vv = 0B else vv = v[i]
    e = CREATE_STRUCT(n[i],vv,e)
;   if ( v[i] eq FLOAT( v[i] ) ) then $
;     e = CREATE_STRUCT(n[i],FLOAT(v[i]),e) else $
;       e = CREATE_STRUCT(n[i],v[i],e)
endfor
return, e
end




function eextra_clean, e
for i = 0, n_tags(e)-1 do begin
    s = e.(i)
    if size(s,/tname) eq 'STRING' then begin
       fix = STREGEX(s,'"(.*)"',/SUBEXPR,/EXTRACT)
       if fix[0] NE '' then e.(i) = fix[1]
       fix = STREGEX(s,'"(.*)"',/SUBEXPR,/EXTRACT)
       if fix[0] NE '' then e.(i) = fix[1]
    endif
endfor
return, e
end

pro eEXTRA_test
input = ''
while (1) do begin
    read, input, PROMPT="IDLVM> " ; get the proc and all keyword arguments
```

```
    cmds = STRTRIM(STRSPLIT( input, ",", /EXTRACT ),2)
    cmd = cmds[0]
    if ( n_elements(cmds) GE 2 ) then $
      e = eEXTRA( cmds[1:*] ) else $
        e = CREATE_STRUCT("IDLVM_NIL",0B) ; null filler
    case cmd of
       'exit': exit
       'stop': stop
       'quit': goto, done
       else: CALL_PROCEDURE, cmd, _EXTRA=e
    endcase
endwhile
done:
end

function eEXTRA, arr
e = eextra_build( arr )
e = eextra_clean( e )
return, e
end
```
