Subject: Re: Recursion in IDL
Posted by alans on Mon, 05 Apr 1993 17:24:25 GMT
View Forum Message <> Reply to Message

Recursion only works on procedures, though. It would be far more useful
for IDL to support recursive *functions*. Unfortunately, this is more
difficult for IDL's parser, I guess, because it has to distinguish between a
function call and an array subscript operation which share the same syntax.
E.G.,

```
function foo, bar
  if (bar eq 1) then $
    return, 1 else $
    return, bar * foo (bar-1)
end

IDL> print,foo(2)
% Variable is undefined: FOO.
% Execution halted at FOO </dev/tty(   1)> .
%      Called from FOO </dev/tty(   2)>.
%      Called from $MAIN$ .
```

Yecch! Does anyone know a way around this? Compile it twice? (Yes,
actually I just tried and it *does* work if you ".run" it twice. (*WOW*) So
I take it all back; IDL *does* support recursive functions, albeit with an
ugly hack. For the record, I'm running IDL 3.0.0 on Sparc SunOs 4.1.1...

RSI, I understand the difficulty involved, but can something be done about
that? I never explicitly ".run" anything; I put all procedures & functions
in my IDL path, and have encouraged my colleagues to do the same...Gee,
think of the CPU-hogging Object-Oriented possibilities with this - functions
which walk structures of structures looking for a particular field names,
recursive "sizeof"-type operations, etc. It boggles the imagination! ;-)
--
Alan J.Stein MIT/Lincoln Laboratory alans@ll.mit.edu

Subject: Re: Recursion in IDL
Posted by thompson on Tue, 06 Apr 1993 13:39:40 GMT
View Forum Message <> Reply to Message

alans@ll.mit.edu (Alan Jay Stein) writes:

> Recursion only works on procedures, though. It would be far more useful
> for IDL to support recursive *functions*. Unfortunately, this is more
> difficult for IDL's parser, I guess, because it has to distinguish between a
> function call and an array subscript operation which share the same syntax.
> E.G.,

> function foo, bar
> if (bar eq 1) then $
> return, 1 else $
> return, bar * foo (bar-1)
> end

> IDL> print,foo(2)
> % Variable is undefined: FOO.
> % Execution halted at FOO </dev/tty(1)> .
> % Called from FOO </dev/tty(2)>.
> % Called from $MAIN$.

> Yecch! Does anyone know a way around this? Compile it twice? (Yes,
> actually I just tried and it *does* work if you ".run" it twice. (*WOW*) So
> I take it all back; IDL *does* support recursive functions, albeit with an
> ugly hack. For the record, I'm running IDL 3.0.0 on Sparc SunOs 4.1.1...

> RSI, I understand the difficulty involved, but can something be done about
> that? I never explicitly ".run" anything; I put all procedures & functions
> in my IDL path, and have encouraged my colleagues to do the same...Gee,
> think of the CPU-hogging Object-Oriented possibilities with this - functions
> which walk structures of structures looking for a particular field names,
> recursive "sizeof"-type operations, etc. It boggles the imagination! ;-)
> --
> Alan J.Stein MIT/Lincoln Laboratory alans@ll.mit.edu

Actually, I can't make your example fail. If I enter "print,foo(2)" then I get the answer "2". I'm using IDL v3.0.0 on SunOS 4.1.2.

Sometimes I've found that IDL can get confused about whether something's a variable or a function when it tries to compile a routine and fails. Then an extra .run seems to solve the problem.

In any case, what you might try is to put a dummy keyword in your function definition. Then the IDL shouldn't have any problems determining that it's a function and not a variable. For example

```
function foo, bar, recursive=recursive
  if (bar eq 1) then $
    return, 1 else $
    return, bar * foo (bar-1,/recursive)
end
```

The recursive keyword above doesn't actually do anything, but it makes it obvious that foo is a function.

Bill Thompson

Subject: Re: Recursion in IDL
Posted by zawodny on Tue, 06 Apr 1993 16:08:01 GMT
View Forum Message <> Reply to Message

In article <sterner.733762629@warper.jhuapl.edu> sterner@warper.jhuapl.edu (Ray Sterner) writes:
> One of the least used features of IDL may be recursion. But it's
> there and works very well.
>
> ...
>
> Ray Sterner sterner@tesla.jhuapl.edu

I agree and wish to post a routine that draws "directory trees" and demonstrates recursion. This routine SPAWNs a lot of child processes and may not be the best way to do this, but it does work.

(there will be a .sig file at the end as well)

```
pro TREE,curdir,maxlevel=maxlevel,file=file $
 ,level=level,flag=flag,tab=tab,stat=stat
;+
; NAME:  TREE
;
; PURPOSE: Draws a subdirectory tree on UNIX systems
;
; CATEGORY: Stuff that noone ever bothers to write
;
; CALLING SEQUENCE:
;  TREE [,dname,maxlevel=n,file=fname]
;
; INPUTS: All inputs are optional
; CURDIR  String variable containing the name of the directory
;   at the root of the tree. (Defaults to '~')
; KEYWORDS:
; MAXLEVEL Integer indicating the number of levels of
;   subdirectories to display.  (Default is to search to
;   a depth of 10 {this is done to avoid infinite loops})
;   Existance of subdirectories beyond MAXLEVEL are
;   indicated by '-->>'.
;
; FILE  String variable specifying the output file name.
;   Special case use of /FILE causes output to go to a file
;   named 'tree.out'. (Default is to output to the screen)
;
```

```
; **** Other Keywords are used to pass data recursively and are, therefore,
; for internal use only and should NOT be utilized by the user.
;
; OUTPUTS:
; May output to either the screen or a file.
;
; COMMON BLOCKS: None
;
; SIDE EFFECTS:  None
;
; RESTRICTIONS:
;   Written to run on UNIX systems
;
;   WARNING! Some patches to operating systems (eg. SUN)
;   have been found to generate infinite loops.  Do not
;   make MAXLEVEL large unnecessarily.
;
;   Really deep searches (MAXLEVEL large) may trash the
;   display when line wrapping is enabled.
;
; PROCEDURE:
; STRAIGHTFORWARD (seems to be the default value of this field).
;
; MODIFICATION HISTORY:
; Written 2/5/93 by J. M. Zawodny, NASA Langley Research Center.
; zawodny@arbd0.larc.nasa.gov
;-

; Somethings are only done at the top level
 if (n_elements(level) eq 0) then begin

 ; Set default values
  if(n_elements(curdir) le 0) then curdir = '~'
  if not keyword_set(maxlevel) then maxlevel = 10
  level = 0

 ; Get the starting directory
  spawn,'cd '+curdir,list
  tab = '      '

 ; Do we write this to a file
  if keyword_set(file) then begin
  ; Get variable type of file
   s = size(file)
  ; Is file a string?
   if(s(1) eq 7) then fname=file else fname='tree.out'
   openw,lun,fname,/get_lun
   flag = lun
```

```
   printf,flag,list
  endif else begin
   flag = 0
   print,list
  endelse
 endif

; Find out what is in the directory
 spawn,'ls -1 -F '+curdir+' | grep "/"',list
 if(list(0) eq '') then begin
  stat = 0
  return
 endif

; Are we beyond desired depth
 if(level ge maxlevel) then begin
  if keyword_set(flag) then printf,flag,tab+'|-->>' $
    else print,tab+'|-->>'
  stat = -1
  return
 endif

; For all subdirectories in this directory
 len = n_elements(list)-1
 for k=0,len do begin
  name = list(k)
  name = strmid(name,0,strpos(name,'/'))
 ; Output a line
  if (k ne len) then tb = tab+'|     ' else tb = tab+'      '
  v = tab+'|-----'+name
  if keyword_set(flag) then printf,flag,v else print,v

 ; Recurse through all subdirectory levels
  tree,curdir+'/'+name,level=(level+1),maxlevel=maxlevel, $
   flag=flag,tab=tb,stat=stat

  if(k ne len) and stat then begin
   tb = tab+'|'
   if keyword_set(flag) then printf,flag,tb $
    else print,tb
  endif

 endfor
 stat = 1

 if keyword_set(file) then begin
  close,lun
  free_lun,lun
```

```
 endif
return
end
```

--

Joseph M. Zawodny (KO4LW) NASA Langley Research Center
Internet: zawodny@arbd0.larc.nasa.gov MS-475, Hampton VA, 23681-0001
Packet: ko4lw@wb0tax.va.usa