Subject: IDL> Looking for routine to list subdirectories.
Posted by Fergus Gallagher on Fri, 05 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Does anyone have a *system independent* procedure to find all the
subdirectories of a given directory. Unix and Windows (as DOS) are easy,
VMS a little bit more tricky. I have no clue about MAC since I don't
have one running IDL, but I would like to write a general routine anyway.

Ideally the Windows listing would not require spawning a sub-shell.

Thanks,

Fergus

```
=================================================
| Fergus Gallagher                              |
| Remote Sensing Applications Development Unit  |
| British National Space Centre                 |
| Monks Wood                                    |
| Huntingdon PE17 2LS / UK                      |
|                                               |
| F.Gallagher@nerc.ac.uk                        |
| http://uh.nmt.ac.uk/bnsc/fgg.html             |
=================================================
```

Subject: Re: IDL> Looking for routine to list subdirectories.
Posted by thompson on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Fergus Gallagher <F.Gallagher@nerc.ac.uk> writes:

> Does anyone have a *system independent* procedure to find all the
> subdirectories of a given directory. Unix and Windows (as DOS) are easy,
> VMS a little bit more tricky. I have no clue about MAC since I don't
> have one running IDL, but I would like to write a general routine anyway.

> Ideally the Windows listing would not require spawning a sub-shell.

The following works in VMS and Unix. The VMS version requires a .COM file,
also appended. I would be interested to see it extended to Windows and MacOS.

Note that the built-in function EXPAND_PATH can be used to do something
similar, but only returns those directories that actually contain procedure
files (or optionally help files). It would be nice if EXPAND_PATH had a
keyword that just made it return all directories regardless of contents.

Bill Thompson

```
 ================================================================
====================
 FUNCTION FIND_ALL_DIR, PATH, PATH_FORMAT=PATH_FORMAT
;+
; Project     : SOHO - CDS
;
; Name        : FIND_ALL_DIR
;
; Purpose     : Finds all directories under a specified directory.
;
; Explanation : This routines finds all the directories in a directory tree
;  when the root of the tree is specified.  This provides the same
;  functionality as having a directory with a plus in front of it
;  in the environment variable IDL_PATH.
;
; Use         : Result = FIND_ALL_DIR( PATH )
;
;  PATHS = FIND_ALL_DIR('+mypath', /PATH_FORMAT)
;
; Inputs      : PATH = The path specification for the top directory in the
;     tree.  Optionally this may begin with the '+'
;     character but the action is the same in any case.
;
; Opt. Inputs : None.
;
; Outputs     : The result of the function is a list of directories starting
;  from the top directory passed and working downward from there.
;  Normally, this will be a string array with one directory per
;  array element, but if the PATH_FORMAT keyword is set, then a
;  single string will be returned, in the correct format to be
;  incorporated into !PATH.
;
; Opt. Outputs: None.
;
; Keywords    : PATH_FORMAT = If set, then a single string is returned, in
;      the format of !PATH.
;
; Calls       : FIND_WITH_DEF
;
; Common      : None.
;
; Restrictions: PATH must point to a directory that actually exists.
;
;  On VMS computers this routine calls a command file,
;  FIND_ALL_DIR.COM, to find the directories.  This command file
;  must be in one of the directories in IDL's standard search
;  path, !PATH.
;
; Side effects: None.
;
; Category    : Utilities, Operating_system.
;
; Prev. Hist. : None.
;
; Written     : William Thompson, GSFC, 3 May 1993.
;
; Modified    : Version 1, William Thompson, GSFC, 3 May 1993.
;  Version 2, William Thompson, GSFC, 6 July 1993.
;   Added sort to spawned command under Unix.
;
; Version     : Version 2, 6 July 1993.
;-
;
 ON_ERROR, 2
;
 IF N_PARAMS() NE 1 THEN MESSAGE, $
  'Syntax:  Result = FIND_ALL_DIR( PATH )'
;
;  Remove any leading + character.
;
 DIR = PATH
 IF STRMID(DIR,0,1) EQ '+' THEN DIR = STRMID(DIR,1,STRLEN(DIR)-1)
;
;  On VMS machines, spawn a command file to find the directories.  Make sure
;  that any logical names are completely translated first.  A leading $ may be
;  part of the name, or it may be a signal that what follows is a logical name.
;
 IF !VERSION.OS EQ 'vms' THEN BEGIN
  REPEAT BEGIN
   IF STRMID(DIR,STRLEN(DIR)-1,1) EQ ':' THEN $
    DIR = STRMID(DIR,0,STRLEN(DIR)-1)
   TEST = TRNLOG(DIR,VALUE) MOD 2
   IF (NOT TEST) AND (STRMID(DIR,0,1) EQ '$') THEN BEGIN
    TEMP = STRMID(DIR,1,STRLEN(DIR)-1)
    TEST = TRNLOG(TEMP, VALUE) MOD 2
   ENDIF
   IF TEST THEN DIR = VALUE
  ENDREP UNTIL NOT TEST
  COMMAND_FILE = FIND_WITH_DEF('FIND_ALL_DIR.COM',!PATH,'.COM')
  SPAWN,'@' + COMMAND_FILE + ' ' + COMMAND_FILE + ' ' + DIR, $
   DIRECTORIES
;
;  On Unix machines spawn the Bourne shell command 'find'.  Remove any trailing
;  slash character from the first directory.
```

```
;
 END ELSE BEGIN
  IF STRMID(DIR,STRLEN(DIR)-1,1) NE '/' THEN DIR = DIR + '/'
  SPAWN,'find ' + DIR + ' -type d -print | sort -', $
   DIRECTORIES, /SH
  TEMP = DIRECTORIES(0)
  IF STRMID(TEMP,STRLEN(TEMP)-1,1) EQ '/' THEN $
   DIRECTORIES(0) = STRMID(TEMP,0,STRLEN(TEMP)-1)
 ENDELSE
;
;  If the PATH_FORMAT keyword was set, then reformat the string array into a
;  single string, with the correct separator.
;
 IF KEYWORD_SET(PATH_FORMAT) THEN BEGIN
  DIR = DIRECTORIES(0)
  IF !VERSION.OS EQ 'vms' THEN SEP = ',' ELSE SEP = ':'
  FOR I = 1,N_ELEMENTS(DIRECTORIES)-1 DO $
   DIR = DIR + SEP + DIRECTORIES(I)
  RETURN, DIR
 END ELSE RETURN, DIRECTORIES
;
 END
 ============================================================
===================
$ VERIFY = 'F$VERIFY(0)'
$!
$!  FIND_ALL_DIR is a VMS command file that supports the IDL routine of the
$!  same name.  It takes two parameters.
$!
$!  P1 Character string denoting where the command file, so
$!   that this routine can call itself.
$!
$!  P2 The next directory to search, one level lower.  If not
$!   passed, then the current directory is used.
$!
$ FILE_SPEC = "*.DIR;0"
$ OLD_DEFAULT = F$LOGICAL("SYS$DISK") + F$DIRECTORY()
$ NEW_DEFAULT = OLD_DEFAULT
$ IF P2 .NES. "" THEN NEW_DEFAULT = P2
$ SET DEFAULT 'NEW_DEFAULT'
$ WRITE SYS$OUTPUT NEW_DEFAULT
$!
$ LOOP:
$ ENTRY = F$SEARCH(FILE_SPEC)
$ IF ENTRY .EQS. "" THEN GOTO DONE
$ OUTLINE = F$PARSE(ENTRY,,,"NAME")
$ NEXT_DIR = NEW_DEFAULT - "]" + "." + OUTLINE + "]"
$ @'P1' 'P1' 'NEXT_DIR' NEXT
$ GOTO LOOP
$!
$ DONE:
$ SET DEFAULT 'OLD_DEFAULT'
$ VERIFY = F$VERIFY(VERIFY)
```