

%TRAVERSE: A Knuth-style Recursive Procedure to Obtain Directory/Filenames

Andy Barnett, CSG, Inc., Raleigh, NC

ABSTRACT

Most early computer languages lacked recursive syntax, but just as D. Knuth showed in his classic texts, using arrays and a few GOTOs solves that problem. Traversing trees, such as the directory/file trees on a typical operating system, or other hierarchical tree structures, is similarly easy to do without explicit recursive functions in SAS® using Knuth's model. The DATA step in %TRAVERSE uses the SAS file functions FILENAME, DOPEN, DNUM, FOPEN, FCLOSE, DCLOSE, and arrays to build a SAS data set containing all the files (accessible by DOPEN and FOPEN) at and below a given "root" directory. The code works on Windows®, and will work on UNIX®, OpenVMS®, and perhaps other systems without much change.

INTRODUCTION

A tree-traversal tool can be quite useful. It's certainly an interesting exercise, but it is also used in the %DIFFTREE (Barnett-2006) macro, and could be used in a variety of problems where hierarchical structures need to be processed. One such problem is the modeling deposition of particles and gases in the lungs as done by the US EPA. Possibly another is working with XML files, which are hierarchical in nature. Many kinds of syntax checkers are hierarchical (see Knuth's differential calculator on p. 342).

BINARY TREE TRAVERSALS

In his series called "The Art of Computer Programming"¹, Donald Knuth describes the three basic binary tree traversals, "preorder", "inorder", and "postorder" (see page 319 of section 2.3.1). These are most easily defined recursively. For a language such as SAS with limited recursion capability, or such as Knuth's MIX assembly language, recursion can be a bit daunting to implement. But his Algorithm T on p. 326 shows how to implement inorder traversal. Preorder and postorder are not much different, and we use Preorder in this paper.

Note the use of stacks and "GOTO" in this algorithm. Without a truly recursive syntax, the use of stacks and GOTOs is necessary.

Here are pseudo-code instructions for the with/without recursive syntax versions of the preorder traversal algorithm:

```
TRAVERSE( subtree_root ) /* with recursion */2
1. Visit the root of this subtree.
2. TRAVERSE( left subtree )
3. TRAVERSE( right subtree )
4. RETURN (unless back at original root, then STOP)

TRAVERSE( subtree_root ) /* without recursion */3
1. Set stack empty, and point P -> subtree_root
2. If P is null, go to step 5.
3. Visit Node(P).
4. Push P onto stack. Set P -> left(node(P)). Go to step 2.
5. Pop top of stack into P. If P is now null STOP.
6. Set P -> right(node(P)). Go to step 2.
```

BUT OUR PC FILE STRUCTURES ARE N-ARY, NOT NECESSARILY BINARY!

Actually, this is ok, since it is very easy to convert any n-ary tree to a binary tree. See Knuth-1997, p. 334, section 2.3.2 for one method. We use this same method here, implicitly.

In the sample file structure below, directories are the blue rounded nodes, and files are the blue rectangular nodes (leaves). The directories/files at/under the Hidden directory have the hidden attribute, under System have the system attribute, and under HiddenSystem have both attributes. The ROOT_HS directory and the files at and below OK are all non-hidden and non-system. This will play an important role in whether or not we can see these files.

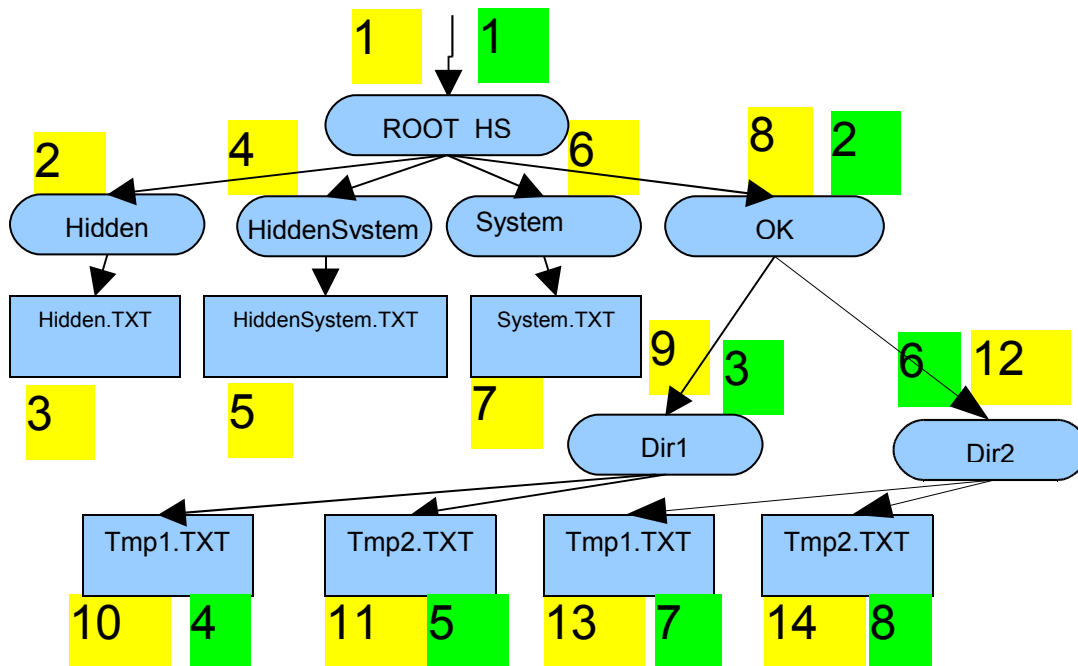
¹ Knuth, 1997, section 2.3.1, pp. 318-334.

² Knuth-1997, p. 319.

³ Adapted from Knuth-1997, p. 320.

The yellow numbers are the preorder sequence of traversal for an algorithm that has access to all files (see the %DIFFTREE paper for this method), and the green numbers are the preorder sequence of traversal for the algorithm in this paper. The SAS functions used to access the files and directories cannot see hidden or system files.

Figure 1: Sample PC file structure:



MACRO PARAMETERS TO %TRAVERSE:

Eight macro parameters control the program, one positional parameter named ROOTPATH, and seven keyword parameters named NROOT, STOP_ABORT, CASE, DEBUG, NAMELENGTH, LIST, and LEVELS. These are described in the header comments in the TRAVERSE macro in Appendix B. The ROOTPATH parameter specifies the pathname to the root of the directory tree to be traversed. NROOT is a numeric tag to suffix the end of the output filename (DIRFILENAMES) in case you want to call %TRAVERSE multiple times (default is null). STOP_ABORT can be either STOP or ABORT, to specify how to halt when errors are discovered; STOP is useful if you are running interactive SAS, and want to be able to save the LOG and OUTPUT windows. ABORT is generally used for batch runs. CASE is set to U to make all paths and filenames uppercase, in order to make comparisons of file/directory names case-insensitive; any other value makes the program case-sensitive. DEBUG is either 1 or 0, where 1 produces extra debugging output. NAMELENGTH is the maximum length for any filename (default is 2000). LIST can be either Y or N, where Y indicates you'd like to save some of the intermediate output on either the PRINT output or in a TXT file (CSV format). Finally, LEVELS indicates how deep the stacks are (e.g., with LINK, this would effectively be about 10); the default is 1000.

SAMPLE CSV OUTPUT:

For purposes of this paper, here is part of the CSV output for the above sample in Figure 1. Comparing these files to the green numbers above shows we have a preorder traversal.

```
"Obs","n","rootpath","subroot","fname","dflag","nroot"
"1",1,"C:\...\ROOT_NORMAL\",".\","D",2007
"2",2,"C:\...\ROOT_NORMAL\",".\","OK",2007
"3",3,"C:\...\ROOT_NORMAL\","OK\","dir1",2007
"4",4,"C:\...\ROOT_NORMAL\","OK\dir1\","tmp1.txt",2007
"5",5,"C:\...\ROOT_NORMAL\","OK\dir1\","TMP2.txt",2007
"6",6,"C:\...\ROOT_NORMAL\","OK\","dir2",2007
"7",7,"C:\...\ROOT_NORMAL\","OK\dir2\","tmp1.txt",2007
"8",8,"C:\...\ROOT_NORMAL\","OK\dir2\","TMP2.txt",2007
```

SOME ADDITIONAL CONSIDERATIONS

1. Recursion is notorious for creating infinite loops. See for example, Knuth's illustration of "Definition, circular" in his index, p. 633.
2. Running this macro and the previous %TRAVERSE (presented in PharmaSUG 2006 as part of %DIFFTREE) on large numbers of files (
3. Filenames with "weird" characters, e.g., "Registration Symbol@.txt", are seen properly by the SAS functions in this %TRAVERSE version, but not in the older version from %DIFFTREE, which shows it as "Registration Symbolr.txt" instead.
4. The speed at which these two programs ran using one experiment is:
TIMESTAMP(PRIOR TO OLDER VERSION)= 19JUL2007:05:58:26.89
TIMESTAMP(AFTER OLDER VERSION)= 19JUL2007:05:58:33.63 = 6.74 secs / 1500 files ~223 files/sec
TIMESTAMP(PRIOR TO THIS VERSION)= 19JUL2007:05:58:33.81
TIMESTAMP(AFTER THIS VERSION)= 19JUL2007:05:59:32.13 = 58.32 secs / 1500 files ~ 26 files/sec
So, the non-recursive, system-dependent version appears about 10x faster with this inconclusive experiment.
5. This macro was written and tested using SAS Learning Edition 4.1, which "includes Base SAS 9.1.3, SAS/STAT, SAS/GRAPH, SAS/QC, SAS/ETS and SAS Enterprise Guide 4.1" and *The Little SAS Book for Enterprise Guide 4.1* (see <http://support.sas.com/le>) . It runs on Windows 2000 and Windows XP (including XP Pro x64).

CONCLUSION

The %TRAVERSE program is a good illustration of how to implement recursion on SAS. The particular application may also find use in any kind of repetitive file handling, especially when the filenames are unknown. %TRAVERSE should run with little or no modification on other platforms, since it is almost entirely SAS-based (with the exception of certain characters/conventions that are system-based).

REFERENCES

1. Barnett-2006: Andy E. Barnett, Paper AP16, DIFFTREE: A Macro to Compare Corresponding Files under Two Directory Trees, PharmaSUG 2006.
2. Knuth-1997: Donald E. Knuth, *The Art of Computer Programming*, 3rd ed., Vol. 1, "Fundamental Algorithms", Boston, MA: Addison-Wesley, 1997.
3. SAS-1999: SAS Institute Inc. 2007. **SAS OnlineDoc® 9.1.3**. Cary, NC: SAS Institute Inc. (see <http://support.sas.com/onlinedoc/913/docMainpage.jsp>).
4. SAS/LE-2007, <http://support.sas.com/le> .

ACKNOWLEDGEMENTS

Thanks to SAS Institute Inc.'s Technical Support for answering several questions during the development of this program; and especially Cindy Puryear of SAS Institute Inc., Kim Sturgen of PPD, Inc., and to Christine Ward of CSG, Inc.

RECOMMENDED READING

Donald E. Knuth, *The Art of Computer Programming*, 3rd ed., Vol. 1, "Fundamental Algorithms", Boston, MA: Addison-Wesley, 1997.

TRADEMARK CITATION

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Andy Barnett, Sr. Programmer Analyst
CSG, Inc.
c/o Christine Ward, Sr. Recruiter
4300 Six Forks Road
Suite 720
Raleigh, NC 27609-5764
919-839-8113
Fax: 919-839-8114
Web: <http://www.csolutionsgroup.com>
Andy's Home Email: aebarnet@nc.rr.com, phone: 919-850-2196

APPENDIX A: SAMPLE INVOCATION:

The following will produce a list of all files and directories under C:\TEMP, in a data set named DIRFILENAMES2007.

```
%traverse( C:\TEMP\, nroot = 2007, stop_abort = STOP, debug = 1, namelength = 1000, list = N )
```

APPENDIX B: %TRAVERSE CODE

```
/*
*****
* PROGRAM NAME:      TRAVERSE_SESUG2007.SAS
* SAS VERSION:      V9.1.3 (SAS/Learning Edition)
* PURPOSE:          Make a list of all files at and below a given directory,
*                   with a DFFLAG=F entry for each file, and a DFFLAG=D entry
*                   for each directory. Provided the root directory exists and
*                   is accessible, then there will be at least one DFFLAG=D
*                   entry in the list.*
* USAGE NOTES:
*   %TRAVERSE( rootpath, <, nroot= > <, stop_abort => <, case=>
*             <, debug = > <, namelength = > <, list = > <, levels = > )
*   where:
*       ROOTPATH      - the path to the root of the subtree,
*       NROOT         - flag to suffix files/data sets with in case
*                       calling %TRAVERSE multiple times
*       STOP_ABORT    - whether to STOP or ABORT (default is STOP)
*       CASE          - U=convert dir/filenames to all upper-case,
*                       else P=preserve case
*       DEBUG         - 0=no debug (default), 1=debug
*       NAMELENGTH    - max filename length (default = 2000)
*       LIST          - N=no output (default), Y=output
*       LEVELS        - depth of stacks (1000 is default)
* INPUT FILES:      all files at and below the ROOTPATH
* OUTPUT FILES:     listing of filenames
*
*MACROS USED/CREATED: %GLB_PRTMPTY
* (see also http://www.lexjansen.com/pharmasug/2006/applicationsdevelopment/ad04.pdf for source
* code for the auxiliary macros)
*   AUTHOR: Andy Barnett
*   DATE CREATED: 18OCT2005
*
*****
* © 2005-2006 Pharmaceutical Product Development, Inc.
* All Rights Reserved.
*****
* MODIFICATION LOG:
* VERSION-YYYYMMDD-INITS - MODIFICATION DESCRIPTION
*****
* V2-20070717-AEB
*   - Andy Barnett, CSG, Inc., modified for SESUG 2007
*   - used by permission of Pharmaceutical Product Development, Inc.
*****
```

```

*/
%macro traverse( rootpath, nroot =, stop_abort = STOP, case = U, debug = 0,
                namelength = 2000, LIST = N, levels = 1000 );
    %local id err dbg kerr;
    %let id = &sysmacroname;
    %let err = E%str(RROR:) &id;
    %let dbg = DEBUG: &id;
    %let kerr = 0;

    %put &id: Begin;
    %put &id: ROOTPATH=&ROOTPATH;;
    %put &id: NROOT=&nroot, STOP_ABORT=&STOP_ABORT, DEBUG=&debug;;
    %put &id: NAMELENGTH=&namelength, LIST=&list;

    %let stop_abort = %upcase(&stop_abort);
    %if %superq(stop_abort) ne STOP %then %let stop_abort = ABORT;
    %let case = %upcase(&case);
    %if %superq(case) ne U %then %let case = P;
    %let namelength = %sysfunc(inputc(&namelength,?best.));
    %if &namelength < 1 or &namelength > 32767 %then %let namelength = 2000;
    %let list = %upcase(&list);
    %if %superq(list) ne N %then %let list = Y;

%*Set pathchars depending on platform;
    %if %index(.WIN.,.&sysscp..) %then %do;
        %local pathchars drivechar diropenchar dirclosechar;
        %let pathchars = :\\;
        %let drivechar = :;
        %let diropenchar = \;
        %let dirclosechar = \;
    %end;
    %else %do;
        %put &err: Un%str(known) operating system (SYSSCP=&sysscp) - aborting;
        %goto err;
    %end;

%*Check that rootpath exists and is accessible;
    data _null_;
        length rootpath errsmsg $32767;
        rootpath = symget( 'rootpath' );
        if filename( 'rootdir', trim( rootpath ) ) ne 0 then link err;
        else do;
            did = dopen( 'rootdir' );
            if did <= 0 then link err;
            rc = dclose( did );
            rc = filename( 'rootdir' );
        end;
        stop;
err:
        errsmsg = sysmsg();
        put 'E' "RROR: &id: Cannot find/access " rootpath= rc= errsmsg;
        call symput( 'kerr', '1' );
        %if %upcase(&stop_abort) = STOP %then stop;
        %else abort abend;
    ;

    run;
    %if &kerr ne 0 %then %goto err;

%*Find absolute ROOTPATH;
    %put &id: ROOTPATH=&rootpath;
    libname rpath "&rootpath" access = readonly;
    %let rootpath = %sysfunc(pathname(rpath));
    libname rpath clear;
    %put &id: ROOTPATH=&rootpath;

%*Add trailing DIRCLOSECHAR if not already present;
    %if %qsubstr(&rootpath,%length(&rootpath),1) ne &dirclosechar
        %then %let rootpath = &rootpath&dirclosechar;

%*Find name of lowest level in ROOTPATH;
    data _null_;
        length rootpath $&namelength;
        rootpath = symget( 'rootpath' );
        %if &diropenchar = &dirclosechar %then rootdir = scanq( rootpath, -1,
"&dirclosechar" );
        %else rootdir = scanq( &rootpath, -1, &pathchars );

```

```

;
if substr( rootdir, length( rootdir ), 1 ) = "&drivechar" then
    %if &diopenchar = &dirclosechar %then rootdir = "&dirclosechar";
    %else rootdir = "&diopenchar&dirclosechar";
;
call symput( 'rootdir', trim( rootdir ) );
put "&id: " rootdir=;
stop;

run;

%*Get subpaths (relative to ROOT) and filenames based on FLSPEC. SUBPATH is the portion of the
path under the ROOT. FLNAME is the filename.filetype without any version number (we will only
look at most recent versions). DFFLAG will be D for directory and F for file;

%*Get the directory and file names for ROOT. We will do this by attempting to DOPEN as a
directory. If this works, we will repeat the process for each directory found. For those files
that do not open with DOPEN, we then know they are files and not directories. The full path will
be composed of ROOTPATH || SUBROOT || FLNAME.
The SUBROOT is the directory path below the ROOTPATH down through the last \.
If the entry is a directory (DFFLAG=D), then SUBROOT is the subpath below the ROOTPATH down to
but not including the last directory level, unless this is null (i.e., we are either at the
ROOTPATH or just one directory below it), in which case SUBROOT is set to .\ (for "here" which
means ROOTPATH). FLNAME will be any remaining subdirectory (never more than one level) with a
trailing \, and in case the entry is the ROOTPATH, then FLNAME=SUBROOT=.\.
For DFFLAG=F, FLNAME is the part of the full path after the last \ (and SUBROOT is the part of
the directory path below ROOTPATH, INCLUDING the last directory level just above this file
entry).
Every directory will have exactly one entry with DFFLAG=D, and every file one entry with
DFFLAG=F.
;

data dirfilenames&nroot ( keep = nroot n rootpath subroot flname dfflag compress = yes );
    length nroot 8 rootpath subroot flname dirname $&namelength dfflag $1;
    retain nroot &nroot rootpath;
    array dids (&levels) 8 _temporary_;
    array istack (&levels) 8 _temporary_;
    array dnums (&levels) 8 _temporary_;
    array dnames(&levels) $&namelength _temporary_;
    array dirnames (&levels) $&namelength _temporary_;
    array flnames (&levels) $&namelength _temporary_;
    retain dids istack . dirnames flnames ' ';
    flname = ' ';
    rootpath = symget( 'rootpath' );
    root = symget( 'root' );
    if root = ' ' then abort abend;
    if substr( root, length( root ), 1 ) ne "&dirclosechar"
    then root = strip( root ) || "&dirclosechar";
    %if &debug %then put "&dbg: CALL DIR: " root=%str(;);
    link dir;
    stop;

dir:
%*Push the current directory/file names onto the stack - first time through DIRNAME=ROOT and
FLNAME=null;
    if ndir = 0 then dirname = root;
    ndir + 1;
    %if &debug %then put "&dbg: DIR: " ndir= dirname=%str(;);
    if ndir > &levels then link errsmsg;
%*See if the DIRNAME+FLNAME is a directory or a file;
    if indexc( flname, "&pathchars" ) > 0 then link errsmsg;
    dirnames(ndir) = dirname;
    flnames(ndir) = flname;
%*Assume DIRNAME+FLNAME is a directory called DNAMES(NDIR);
    if dirname ne ' ' then do;
        if substr( dirname, length(dirname), 1 ) ne "&dirclosechar"
        then dnames(ndir) = trim( dirname ) || "&dirclosechar" || flname;
        else dnames(ndir) = trim( dirname ) || flname;
    end;
    else dnames(ndir) = flname;
    fref = 'F' || put( ndir, z7. );
    rc = filename( fref, trim( dnames(ndir) ) );
    %if &debug %then put "&dbg: DIR: PUSH: FILENAME: " rc= ndir= dnames(ndir)=%str(;);
    if rc = 0 then do;
%*Attempt to open as a directory;
        dids(ndir) = dopen( fref );
        %if &debug %then put "&dbg: DOPEN: " dids(ndir)= dnames(ndir)=%str(;);
        if dids(ndir) > 0 then do;

```

```

*Success - DNAMES(NDIR) is a directory;
        dfflag = 'D';
        dirname = dnames(ndir);
        flname = ' ';
        link output;
*Traverse this DNAMES(NDIR) directory by selecting each FLNAME in turn,
and testing each file like we did for this DNAMES(NDIR);
        dnums(ndir) = dnum( dids(ndir) );
        istack(ndir) = 1;
        do while ( istack(ndir) <= dnums(ndir) );
            dirname = dnames(ndir);
            flname = dread( dids(ndir), istack(ndir) );
            if index( upcase( flname ), upcase( dirname ) ) = 1
            then flname = substr( flname, length( dirname ) + 1 );
            %if &debug %then
                put "&dbg: DREAD: PUSH: " ndir= dids(ndir)=
                    istack(ndir)= dnums(ndir)= dirname=
                    flname=%str();
            goto dir;

dirreturn:
            istack(ndir) + 1;
            %if &debug %then
                put "&dbg: DREAD: POP: " ndir= dids(ndir)=
                    istack(ndir)= dnames(ndir)= flname=%str();
            end;
            rc = dclose( dids(ndir) );
            %if &debug %then put "&dbg: DCLOSE: " rc= ndir= dids(ndir)=
                dnames(ndir)=%str();
            end;
            else link testfile;
        end;
        else link testfile;
        fref = 'F' || put( ndir, z7. );
        rc = filename( fref );
        %if &debug %then put "&dbg: FILENAME CLEAR: " rc= ndir= dirnames(ndir)=
            flnames(ndir)=%str();
        if rc ne 0 then link errsysmsg;
*Pop the last directory we were at;
        ndir + (-1);
        %if &debug %then put "&dbg: " ndir=%str();
        if ndir > 0 then do;
            dirname = dirnames(ndir);
            flname = flnames(ndir);
            %if &debug %then put "&dbg: DIR: POP: " ndir= dirname= flname=%str();
            goto dirreturn;
        end;
        %if &debug %then put "&dbg: DIR RETURN"%str();
        return;

testfile:    dfflag = 'F';
*Not a directory;
        dirname = dirnames(ndir);
        flname = flnames(ndir);
        if dirname ne ' ' then do;
            if substr( dirname, length( dirname ), 1 ) = "&dirclosechar"
            then fname = trim( dirname ) || flname;
            else fname = trim( dirname ) || "&dirclosechar" || flname;
        end;
        else fname = flname;
        %if &debug %then put "&dbg: " dfflag= ndir= dirname= flname= fname=%str();
        fref = 'F' || put( ndir, z7. );
        rc = filename( fref, trim( fname ) );
        if rc ne 0 then link errsysmsg;
        else do;
            fid = fopen( fref );
            if fid > 0 then do;
                %if &debug %then put "&dbg: FILE: FOPEN: " ndir= dfflag= fname=
                    fid=%str();
                link output;
                rc = fclose( fid );
            end;
            else link errsysmsg;
            rc = filename( fref );
        end;
        return;

```

```

output:      n + 1;
             length fullname $&namelength;
             if dirname = ' ' then dirname = '.'\';
             else if substr( dirname, length( dirname ), 1 ) ne '\'
               then dirname = strip( dirname ) || '\';
             fullname = strip( dirname ) || flname;
*Strip off the ROOTPATH;
             if index( upcase( fullname ), upcase( strip( rootpath ) ) ) ne 1 then abort abend;
             subroot = substrn( fullname, lengthn( strip( rootpath ) ) + 1 );
             if subroot = ' ' then subroot = '.'\';
*Find the filename (portion after last \);
             flname = scanq( strip( subroot ), -1, '\' );
             if &debug then put "OUTPUT: " dfldflag= fullname= / subroot= flname=;
             if substrn( strip( subroot ),
               lengthn( strip( subroot ) ) - lengthn( flname ) + 1 ) ne flname
               then do;
*The FLNAME is actually a directory level;
               if dfldflag ne 'D' then abort abend;
               flname = strip( flname ) || '\';
             end;
             else do;
*The entry is a file;
               if dfldflag ne "F" then abort abend;
             end;
             subroot = substrn( subroot, 1, lengthn( subroot ) - lengthn( flname ) );
             if subroot = ' ' then subroot = '.'\';
             if &debug then put
               / 'DEBUG: OUTPUT: ' n= dfldflag= dirname=
               / ".....: " flname= subroot=;
             output;
             return;

errsysmsg:   errsysmsg = sysmsg();
             link err;
             return;

err:         put "E" "RROR: &id: " _all_;
             call symput( 'kerr', '1' );
             stop;

run;

%if &kerr %then %goto err;

%let nowtime = %sysfunc(putn(%sysfunc(date()), yymmddn8.));
%let nowtime = %sysfunc(compress(&nowtime._%sysfunc(putn(%sysfunc(time()), time8.)), :-));
ods csv file = "traverse_&nowtime..csv";
proc print uniform width= min data = dirfilenames&nroot;
    var n rootpath subroot flname dfldflag nroot;
run;
ods csv close;

%if %qupcase(&list) = Y %then %do;
    %glb_prtempty( dirfilenames&nroot, printnull = Y,
        nobyline = rootpath subroot,
        reportby = rootpath subroot,
        reportcol = flname nroot dfldflag n,
        reportwid = 80      8      1      12,
        reportflow = flname,
        reportorder = flname nroot )
%end;
%goto fin;

%err:      %put &err: Aborting ...;
           %put &err: ... ROOTPATH=&rootpath;;
           %put &err: ... NROOT=&nroot, STOP_ABORT=&stop_abort, DEBUG=&debug, NAMELENGTH=&namelength;
           %if %qupcase(&stop_abort) = STOP %then %goto fin;
           %else e%str(ndsas);

%fin:      %put &id: End;

           %put global;
           %put local;
%mend traverse;

```