Paper 87-28

The power of recursive SAS® macros - How can a simple macro do so much ?

John H. Adams, Boehringer Ingelheim, Ridgefield, CT

ABSTRACT

A recursive macro is a perfect solution if you need to apply a 'fixed process' in a nested application. Using regular macros, you would need to know all the dimensions or levels of nesting in order to set up the appropriate 'nested do loops', resulting in a long complex macro program. With recursive macros, however, you do not need to know the levels of nesting up front, as the macro can call itself over and over again , as needed, to apply a given process.

This paper describes a simple recursive macro, running on a Windows platform, that explores all the branches of a Unix file/directory tree, starting from a given entry point on down. The macro reads each directory , gets a list of dataset names and adds it to the master list. It then opens up each subdirectory it found and reads it. Ultimately, it will have traveled all branches to their ends in order to assemble a full list of datasets (or any other selected file type).

INTRODUCTION

Did you ever have write an application program that had to do a very repetitive task over many iterations? Do you use nested macros, or maybe data steps with arrays and extensive looping? There are probably several solutions. Each of them would require you to have some idea about the dimensions and possible levels of nesting needed if performing the task.

Let's take this example: You have been asked to assemble a full list of all SAS datasets on your Unix server , given an appropriate starting path to a directory. Each directory can have datasets, links , subdirectories and other type of files. So, here's the process: 1. read the directory, 2. keep the dataset names and 3. loop through each subdirectory in this directory. For each subdirectory you repeat the process, until there are no more subdirectories are found . Now , you're at the end of this branch and you have to return to the last point were this branch started and search the next branch. (see figure 1 for a sample (NFS) directory tree).

You can quickly see that this approach would require a tremendous amount of nested looping and could become quite complex. Additionally, you may have to a consider a pre-processing step to determine the amount of nesting needed.

There must be a better way!

THE ELEGANT SOLUTION

The process seems so simple. So how can we simplify our program? A recursive macro, of course. A recursive macro is one that calls itself over and over to perform a specific task.

Recursive macros may seem intimidating, difficult to understand and implement to most programmers. They are, however quite simple to understand and not difficult to program – if you follow one simple rule: localize internal macro variables. This prevents subsequent calls of the macro from trashing macro variable values of previous calls (threads that are still active). When a branch end is encountered, the previous call is restored with all of it's original macro values.

THE MAIN MACRO (%UNIX_LST)

In the above example, the main macro (%unix_lst) runs in a local SAS session that has a connection to a remote Unix server (using **SAS®** CONNECT). See Figure 3 for a look at the code.

This main macro has 2 sub macros defined :

1. a recursive macro (%process)
2. a macro (%trans) that reads a directory on the remote session.

HOW IT WORKS

The main macro makes only one call to the %process macro.

The %process macro calls %trans to read the current directory and then calls itself again and again for each subdirectory it found, each time updating the list dataset. When it reaches the end of a branch, it travels back to the top of that branch and goes down the next branch.

Eventually, all branches will have been searched and the final dataset will contain the names of the datasets found , their size, their paths, owners, etc. This master list dataset be directly used in other applications or you can use Proc Report to produce a listing. NOTE: %unix_lst can also look for other file types besides datasets.

So, this recursive approach allows us to search the directory tree to any depth level. We don't even have to know how complex the tree is up front. It is almost like an independent agent or robot that knows what to do. That's the power and the beauty of recursive macros!

HOW IT IS USED

The %unix_lst macro call is very simple :

```
%UNIX_LST(subdir=Y,          /*search structure below?*/
          path=$RXC_SAS_VIEW/nevir,
                             /*sample starting directory*/
          type='sd02' snx02',
                              /*file extensions selected*/
          typname=DATASET,
                             /*name the selected type*/
          outdata=NEVIR );
                             /* name of final list dataset */
```

See figure 2 for the log and the output list dataset of a sample call. Each line in the partial output shows the path of the directory, name of the file, server id, and the various file attributes (i.e. owner name, permissions, dates, file size etc.)

UNDERNEATH THE COVERS

In figure 3 you can see the actual macro code. It's fairly well commented in order to explain the functions of the various sections. Notice the heavy use of SQL to create the various control macro variables throughout.

The remote %trans macro shows how to read the file, which is the re-directed output of a unix command (x ll > $HOME/temp.txt), with a data step (note: 'll' is a Unix command that reads the directory). An alternative would been to use a 'pipe' to read the results of the 'll' command directly.

For those of you that always enjoy finding new techniques, look at the use of the local macro '&lstr' to dynamically communicate with the remote session. It is dynamically build (locally) to contain all the code you wish to submit remotely and then executed with the '%unquote(&lstr) ;' statement.

Also interesting may be the use of the '&selins1-&selins5' macro variables as templates to build dynamic SQL querys to fill the '&ndc &dcn &nds' macro variables with their new values for the current directory. The '&ndc' macro variable actually controls the recursive calling of the %process macro.

FIGURE 1 - A SAMPLE DIRECTORY STRUCTURE ON UNIX

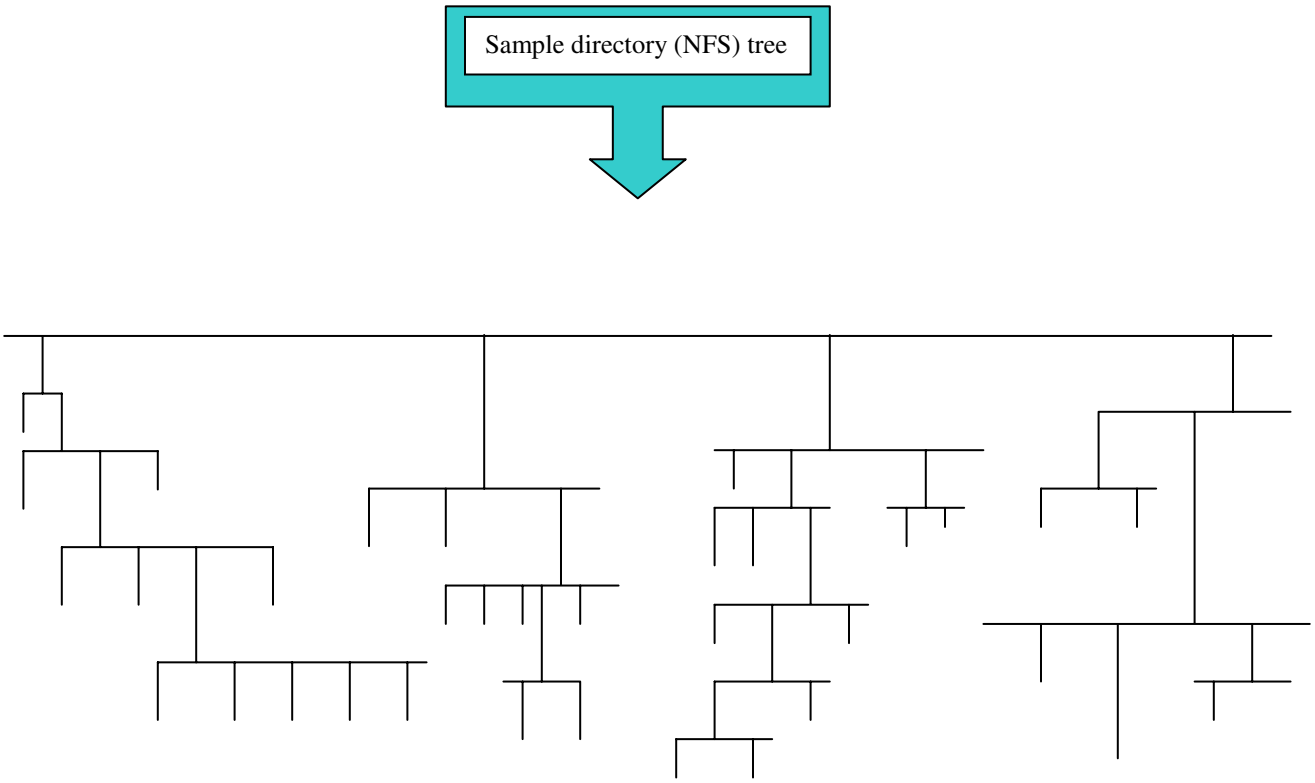


FIGURE 2- OUTPUT FOR SAMPLE USE OF THE MACRO

THE MACRO CALL:

```
%unix_lst(subdir=Y,path=$RXC_SAS_VIEW/nevir,
          outdata=NEVIR );
```

THE LOG:

(note the # of datasets(#D/S) and # of directories(#DIRS) at each node):

Level 0: STARTING PATH=$RXC_SAS_VIEW/nevir #D/S=0 #DIRS=3

Level 0_1 PATH=$RXC_SAS_VIEW/nevir/1036 CTR #D/S=0 #DIRS=2

Level 0_1_1 PATH=$RXC_SAS_VIEW/nevir/1036 CTR/current #D/S=0 #DIRS=0

Level 0_1_2 PATH=$RXC_SAS_VIEW/nevir/1036 CTR/test #D/S=7 #DIRS=0

Level 0_2 PATH=$RXC_SAS_VIEW/nevir/E_Sub #D/S=0 #DIRS=2

Level 0_2_2 PATH=$RXC_SAS_VIEW/nevir/E_Sub/test #D/S=13 #DIRS=4

Level 0_2_2_1 PATH=$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037 #D/S=8 #DIRS=0

Level 0_2_2_2 PATH=$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1038 #D/S=8 #DIRS=0

Level 0_2_2_3 PATH=$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1046 #D/S=11 #DIRS=0

Level 0_2_2_4 PATH=$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1090 #D/S=20 #DIRS=0

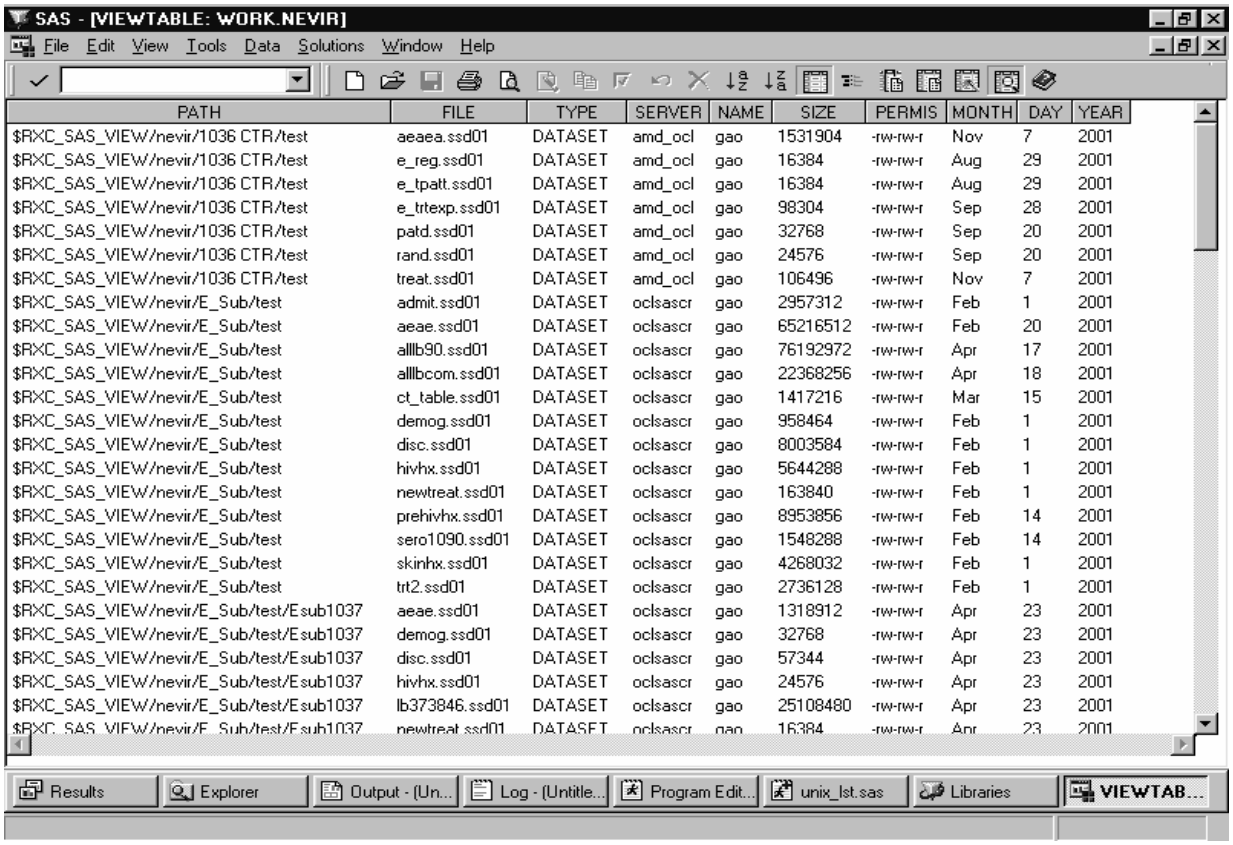
```
Level 0_3
PATH=$RXC_SAS_VIEW/nevir/SR182 #D/S=0
#DIRS=2

Level 0_3_1
PATH=$RXC_SAS_VIEW/nevir/SR182/current
#D/S=0 #DIRS=0

Level 0_3_2
PATH=$RXC_SAS_VIEW/nevir/SR182/test #D/S=47
#DIRS=0
```

NOTE: Table WORK.NEVIR created, with 114 rows and 10 columns.

THE LIST DATASET



PATH	FILE	TYPE	SERVER	NAME	SIZE	PERMIS	MONTH	DAY	YEAR
$RXC_SAS_VIEW/nevir/1036 CTR/test	aeaea.ssd01	DATASET	amd_ocl	gao	1531904	-rw-rw-r	Nov	7	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	e_reg.ssd01	DATASET	amd_ocl	gao	16384	-rw-rw-r	Aug	29	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	e_tpatt.ssd01	DATASET	amd_ocl	gao	16384	-rw-rw-r	Aug	29	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	e_trtexp.ssd01	DATASET	amd_ocl	gao	98304	-rw-rw-r	Sep	28	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	patd.ssd01	DATASET	amd_ocl	gao	32768	-rw-rw-r	Sep	20	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	rand.ssd01	DATASET	amd_ocl	gao	24576	-rw-rw-r	Sep	20	2001
$RXC_SAS_VIEW/nevir/1036 CTR/test	treat.ssd01	DATASET	amd_ocl	gao	106496	-rw-rw-r	Nov	7	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	admit.ssd01	DATASET	oclsascr	gao	2957312	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	aeae.ssd01	DATASET	oclsascr	gao	65216512	-rw-rw-r	Feb	20	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	alllb90.ssd01	DATASET	oclsascr	gao	76192972	-rw-rw-r	Apr	17	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	alllbcom.ssd01	DATASET	oclsascr	gao	22368256	-rw-rw-r	Apr	18	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	ct_table.ssd01	DATASET	oclsascr	gao	1417216	-rw-rw-r	Mar	15	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	demog.ssd01	DATASET	oclsascr	gao	958464	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	disc.ssd01	DATASET	oclsascr	gao	8003584	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	hivhx.ssd01	DATASET	oclsascr	gao	5644288	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	newtreat.ssd01	DATASET	oclsascr	gao	163840	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	prehivhx.ssd01	DATASET	oclsascr	gao	8953856	-rw-rw-r	Feb	14	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	sero1090.ssd01	DATASET	oclsascr	gao	1548288	-rw-rw-r	Feb	14	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	skinhx.ssd01	DATASET	oclsascr	gao	4268032	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test	trt2.ssd01	DATASET	oclsascr	gao	2736128	-rw-rw-r	Feb	1	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	aeae.ssd01	DATASET	oclsascr	gao	1318912	-rw-rw-r	Apr	23	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	demog.ssd01	DATASET	oclsascr	gao	32768	-rw-rw-r	Apr	23	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	disc.ssd01	DATASET	oclsascr	gao	57344	-rw-rw-r	Apr	23	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	hivhx.ssd01	DATASET	oclsascr	gao	24576	-rw-rw-r	Apr	23	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	lb373846.ssd01	DATASET	oclsascr	gao	25108480	-rw-rw-r	Apr	23	2001
$RXC_SAS_VIEW/nevir/E_Sub/test/Esub1037	newtreat.ssd01	DATASET	oclsascr	gao	16384	-rw-rw-r	Apr	23	2001

FIGURE 3- THE MACRO CODE

```
*=======================================*
*   Source code for UNIX_LST  macro     *
*        Program  by John H. Adams      *
*         06/19/2002   version 1.1      *
*        (Using a Recursive Macro       *
 =======================================*;
/*=======================+==============*
 *  unix_lst : Reads the directory of a  *
 *           unix folder and retrieves the names *
 *           (and attributes)  of selected file types, *
 *           ie. file types (extension  of files). It will also *
 *           continue to look (optional)  at all sub- *
 *           directories   levels below the initial   folder *
 *           and retrieve those  file names. The list and *
 *           attributes will be left in the dataset named *
 *           by outdata argument          *
 *  usage : unix_lst(path=,type=,  typname=,outdata=, *
 *                   subdir= );           *
 *  assumptions: 1.You are connected to  a server *
 *                2 You have a libref for  RWORK *
 *                  (remote WORK    folder) *
 *---------------------------------- ----------------------------------*/
 * arguments : path   = path to initial   folder (req) *
*                type   = extension(s) of   selected *
 *                   files(opt)           *
 *                defaults('ssd01' 'snx01' 'sas7bdat' *
*                    'sas7bndx')          *
 *              typname= Type Name for selected *
 *                   files(opt)  default (DATASET'*
*              outdate = ouput dataset name *
 *                   containg filelist(req) *
 *              subdir = Search sub-directories? *
 *                   [ N or Y] (opt)      *
 *                   default(N)           *
 *=======================================*/

%macro unix_lst(subdir=N,path=, outdata=,
                type='ssd01' 'snx01'    'sas7bdat' 'sas7bndx',
                typname=DATASET)
        /des='UNIX folder(s) File Listing Macro';

   %local wd1 wd2 wd3 wd4 wd5 i j k dirsw targ tpath
          tpath1 tpath2 tpath3 tdsout ttype ttypname
          lstr targ dirsw selins1 selins2 selins3 selins4
          nds ndc dcn nds1 ndc1 dcn1 nds2 ndc2
          dcn2 nds3 ndc3 dcn3 ;

   options nomprint nonotes;



Rsubmit;
/* Create remote macro %trans to read file of directory */

  options nomprint nosource nonotes;

%macro trans(path=,outdata=, typname=DATASET,
             type='ssd01' 'snx01''sas7bdat' 'sas7bndx')
        /des='UNIX Directory read macro';

    x cd &path; x ll > $HOME/temp.txt;
    filename temp "$HOME/temp.txt";

    data &outdata(drop=file1 file2);
       *** create dirlist dataset ***;
     length  file1 file2 file3 file4 file5  type $50
             file $100 path  $100;;
     infile temp lrecl=100 firstobs=2 Missover;
      input  permis $ num1 $ name $ server $ size  $ month $
             day $ year $ file1 $ file2 $ file3 $ file4 $ file5 $;

      file =   trim(file1)||' '||trim(file2)||' '||trim(file3)||' '||trim(file4)
             ||' '||trim(file5);
      type=' ';  path = "&path";

      if substr(permis,1,1)='d' then   type='DIRECTORY';
      else if substr(permis,1,1)='-'  and trim(scan(file,2,'.'))
        in(&type)    then type="&typname";

      if type ^=' ' then output;
   run;
     x rm $HOME/temp.txt;
%mend trans;                    /* End of remote macro  */


endrsubmit;



       /* CREATE THE RECURSIVE PROCESS MACRO */

%macro process(lvl=,ttpath=,dnames=, ndnames=,permis=);

     %local ndc nds dcn m wd1 wd2 perm ;
  %do m=1 %to &ndnames;
    %let ndc=0;  %let nds=0;  %let dcn=;
    %let wd1=%scan(&dnames,&m,%str(~));
    %let wd2=%scan(&permis,&m,%str(~));

    %if %nrbquote(%substr(&wd2,5,1))= %nrbquote(-)  %then
        %put NOTE: No permission to read sub- directory "&wd1";

   %else %if
          %index(%upcase(&wd1),%str(MISSING))>0
      or   %index(%upcase(&wd1),%str(FILES))>0        %then
       %put  NOTE: Sub-directory "&wd1" not explored;

   %else %do;
      %let tdsout=tmp;
      %let tpath=&ttpath%str(/&wd1);  %put;
      %let lstr=%str(rsubmit;)
                %nrbquote(%)trans(%unquote(&targ))
                %str(; endrsubmit;);

      %unquote(&lstr);                  /* Submit to server */

       proc sql noprint;
        %unquote(&selins3;
                  &selins1 :  ndc &selins4 ;
                  &selins2 :  dcn &selins4;
                  &selins1 :  nds  &selins5) ;
       quit;

    %put Level &lvl._&m PATH=&tpath #D/S=&nds #DIRS=&ndc;

    %if &nds >1 %then %do;
      proc append
          base=rwork.&outdate
          data=rwork.&tdsout
          (where=(type="&typname"));
             run;
    %end;

    %if &ndc >0 %then
       %process(lvl=&lvl._&m, ttpath=&tpath,dnames=&dcn,
                ndnames=&ndc,permis=&perm);
  %end;
 %end;

%mend process;                  /* End of recursive  macro  */
```

```
            /*  building %trans argument template  */
   %let
targ=path=%nrbquote(&)%str(tpath,type=)%nrbquote(&)%str(ttyp
e,outdata=);

  %let
targ=&targ%nrbquote(&)%str(tdsout,typname=)%nrbquote(&)ttyp
name;

            /*  building sql query templates */
  %let selins1=select count(distinct file) into ;
  %let selins2=select distinct file into ;
  %let selins3=select permis into:perm separated by '~' from
     (select distinct file,permis from
      rwork.%nrbquote(&)tdsout where  type='DIRECTORY');
  %let selins4=separated by '~' from
            rwork.%nrbquote(&)tdsout where  type='DIRECTORY';
  %let selins5=separated by '~' from
            rwork.%nrbquote(&)tdsout where  type="&typname";


  %let ndc0 =0;  %let nds0=0;  %let dcn0=;

          /*Do Initial folder */
  %let tpath=&path; %let tdsout=&outdata;
  %let ttype=&type; %let ttypname=&typname;



%let lstr=%str(rsubmit;)%nrbquote(%)trans(%unquote(&targ))
%str(; endrsubmit;);

%unquote(&lstr);                        /* Submit to server */

              /* Get Initial folder Info */
   proc sql noprint;
     %unquote(&selins3; &selins1 : ndc0
       &selins4 ; &selins2 : dcn0
       &selins4 ; &selins1 : nds0
              &selins5);
   quit;

%if %upcase(%substr(&subdir.NO,1,1))^=N   %then %do;

     /* Do levels below Initial folder */
    %let i=1;   %let dirsw=1;
    %let tdsout=&outdata;

    %put Level 0: STARTING PATH=&tpath   #D/S=&nds0
        #DIRS=&ndc0;

  %process(lvl=0,ttpath=&tpath,
          dnames=&dcn0,
          ndnames=&ndc0,
          permis=&perm);
%end;

  options nomprint notes;

    /* Final re-order of vars, move to WORK */
  proc sql noprint;
    create table &outdata as
      select path,file,type,server, name,size,
               permis,month, day,year
      from rwork.&outdata
      where type="&typname";
  quit;

%mend unix_lst;     /*end of main macro*/
```

CONCLUSION

Recursive macros can make an open ended process application extremely easy. The macro presented here, for example, can travel any number of levels down a NFS directory tree and perform a given task. This task could be as simple as assembling a list of certain type of files, to interrogating and / or moving data to other locations.

CONTACT INFORMATION

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

John H Adams
Boehringer Ingelheim
900 Ridgebury Road
Ridgefield, CT 06877-0368
Work Phone: 203-778-7820
Fax: 203-798-4282
E-mail: jadams@boehringer-ingelheim.com