AP16
DIFFTREE: A Macro to Compare Corresponding Files under Two Directory Trees

Andy Barnett, PPD, Morrisville, NC

ABSTRACT

DIFFTREE is a macro that compares two directory trees with each other to ensure they each contain the same files, and that the files have the same data. The procedure works on both OpenVMS and Windows 2000. The program is particularly helpful when making CDs or DVDs to verify that the CDs/DVDs match the original files on a hard drive or other location. DIFFTREE reports whether or not each file from a corresponding pair is (1) the same as its correspondent in the other subdirectory, (2) different, or (3) non-existent. It uses SAS® functions like FILENAME, DOPEN, FOPEN, FREAD, FGET, FRLEN, FCLOSE, DCLOSE, etc. to check both trees using two main DATA steps, reading the files in binary mode up to 32,767 bytes at a time.

INTRODUCTION

The process requires two basic steps, (1) to traverse each tree to find all nodes therein, and (2) to compare corresponding nodes. The first step, handled by a TRAVERSE macro, requires the pathnames to the top (root) of each tree. This information is used to find all nodes below each root. A node is a "stem node" if it is a directory (meaning it can have subdirectories or files within it), or a "leaf node" if it is a file. The second step (DIFFTREE), compares all corresponding nodes in the two root directories and their subdirectories, classifying each pair as either different, same, or missing (one or the other tree does not have the node).

Step 1: Getting the List of Directory and File Names (Nodes) – TRAVERSE macro

The original version of the TRAVERSE macro used a recursive algorithm to traverse each tree, using two GOTOs and a set of stacks (arrays) to handle the recursion. The GOTO/stack method was needed because the SAS LINK statement has limited depth. The algorithm would start at the top of each root, attempt to open that node as a directory with DOPEN, and if it succeeded, then push the directory on the stack, and proceed down to the first directory/file below it indicated with DNUM/DREAD. If a node could not be opened via DOPEN, then it must be a file. Eventually, all nodes are visited, and there is nothing left on the stack to check. The recursion is an interesting exercise, and was efficient, but has drawbacks surrounding the DNUM/DREAD functions. Namely, DNUM/DREAD does not return Windows files with either the "system" or "hidden" attributes, and they only return the most recent version of files on OpenVMS.

To get around this, FILENAME PIPE was investigated, and this was very slow on OpenVMS. So, the latest method for TRAVERSE is to use an X-statement to submit an operating system command to list the filenames to a temporary file in the SAS WORK directory (see the FILENAME DIR… statements in the listings below). Some minor problems were encountered here. In Windows, certain filenames are not listed correctly by the DIR command (e.g., a filename containing a "®" symbol shows up instead as an "r", and the subsequent comparison cannot open the file to read it, since it is essentially the wrong name). And in OpenVMS, blank lines are generated on occasion when a particularly long filename is listed, but these are simply discarded without any problem.

Step 2: Comparing Corresponding Nodes – DIFFTREE macro

Once the directory and file names (nodes) of each tree are collected by TRAVERSE, the list is divided into those nodes that are present in each tree (MATCH), and those that are in one but missing from the other (MISSING), using PROC SQL. Then, for those nodes that match in both trees, a comparison of file contents is done using a 2nd DATA step. Files are opened for binary reads using FOPEN, taken 32,767 bytes at a time (the last read will be less than this if the file is not a multiple of 32,767 bytes) using FREAD, FGET, and FRLEN; and compared. Any differences in

records, end-of-file position, or FRLEN, signal that the files differ. (Note that this method still works even if the actual record length of the file exceeds 32767.)

Macro Parameters to DIFFTREE:

Eight macro parameters control the program, two positional parameters named ROOTPATH1 and ROOTPATH2, and six keyword parameters named STOP_ABORT, WHERE, CASE, DEBUG, NAMELENGTH, and LIST. These are described in the header comments in the DIFFTREE macro in Appendix D. The ROOTPATH1 and ROOTPATH2 parameters specify the pathname to the root of each directory tree. 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. WHERE is used to limit the output (e.g., one can skip listing files that are the SAME, by using "COMPARE NE 'SAME'"). 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 (note that this does not affect the data within files). DEBUG is either 1 or 0, where 1 produces extra debugging output. NAMELENGTH is the maximum length for any filename (default is 2000). And 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).

An example

For an illustration, assume we have two subtrees below the current directory ("."), each starting at TEST1 and TEST2. The test directory and file names are given below, where directory names are shown with a trailing "\":

```
.\TEST1\                                    .\TEST2\
BOTH\                                       BOTH\
BOTH\BOTH_DIFF.doc                          BOTH\BOTH_DIFF.doc
BOTH\BOTH_DIFF.TXT                          BOTH\BOTH_DIFF.TXT
BOTH\BOTH_SAME.doc                          BOTH\BOTH_SAME.doc
BOTH\BOTH_SAME.TXT                          BOTH\BOTH_SAME.TXT
BOTH\DIFF\                                  BOTH\DIFF\
BOTH\DIFF\CPORT.CPT                         BOTH\DIFF\CPORT.CPT
BOTH\DIFF\CPORT.LOG                         BOTH\DIFF\CPORT.LOG
BOTH\DIFF\CPORT.SAS                         BOTH\DIFF\CPORT.SAS
BOTH\EMPTY1_NOT_IN_TEST2\
BOTH\EMPTY_BOTH\                            BOTH\EMPTY_BOTH\
NOT_IN_TEST2\                               NOT_IN_TEST1\
NOT_IN_TEST2\COMPSORT_20051119.doc          NOT_IN_TEST1\COMPSORT_20051119.doc
NOT_IN_TEST2\glb_coalesce.sas               NOT_IN_TEST1\glb_coalesce.sas
```

These are setup so that TEST1 and TEST2 each have the files in the BOTH subdirectory, but some are different (DIFF) and others are identical (SAME). There are also files/directories in each tree that are not in the other. E.g., directory NOT_IN_TEST2 is in TEST1, but not in TEST2. And even though the two files under TEST1\NOT_IN_TEST2 and TEST2\NOT_IN_TEST1 are identical, they are not corresponding since they are in different subtrees. Thus, these are considered "MISSING". In particular, The file BOTH\DIFF\CPORT.CPT is a CPORT transport file that was copied to one test area via ASCII FTP, and the other via BINARY FTP, and it shows up as DIFF in the example output.

When the files are all processed, they are listed in one of two reports. The first report (MISSING) shows all nodes that are present in one tree but not in the other. The 2nd report lists the remaining pairs of corresponding nodes by COMPARE SUBPATH FILENAME DFTYPE, where COMPARE is either SAME or DIFF. Normally, we just exclude the COMPARE=SAME results (by way of the line "WHERE=COMPARE NE 'SAME'" in the parameter file). But if this information is needed, one can use "WHERE=". (See Appendices A and B for output from this test case.)

CONCLUSION

The DIFFTREE program is fairly quick and painless in classifying files as “MISSING”, “DIFF”, or “SAME”. For tasks (such as burning a CD or DVD) where we have (1) very large files, (2) large numbers of files, or (3) files of various types (ZIP, DOC, TXT, SAS, etc.), and we want to ensure they are copied correctly, this is an easy method to use.

REFERENCES

SAS Institute Inc., SAS OnlineDoc®, Version 8, Cary, NC: SAS Institute Inc., 1999.

ACKNOWLEDGMENTS

Thanks to SAS Institute Inc.’s Technical Support for answering several questions during the development of this program, and to Kim Sturgen, Ed Lunk, Hunter Everton, and Cecilia Mauldin for their support and advice.

CONTACT INFORMATION

For more information (including electronic copies of this program) or to send comments/questions, contact the author at:

Andy Barnett, Sr. Programmer Analyst
PPD
3900-N Paramount Pkwy.
Morrisville, NC 27560
Work Phone: 919-462-4010
Fax: 919-379-3285
Email: Andy.Barnett@rtp.ppdi.com
Web: www.ppdi.com

Appendix A: Sample Invocation

```
/*NOTE: Several auxiliary macros are in MACROS.SAS – these are shown in Appendix E*/
%include 'macros.sas';
%init;

%difftree(

C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST1,
C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST2,
debug = 0, list = N

)
```

Appendix B: Sample Output (NOTE: some spacing removed to save space)

PRINT: DSN=MISSING (), NOBS=10, OBS=MAX
MISSING: These files were found without corresponding files in other ROOTPATH
ROOTPATH1=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST1\
ROOTPATH2=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST2\

subroot	flname	nroot	dfflag	n
.\	NOT_IN_TEST1	2	D	3
	NOT_IN_TEST2	1	D	3
BOTH\	EMPTY1_NOT_IN_TEST2	1	D	10
	NOT_IN_TEST2	1	D	12
BOTH\NOT_IN_TEST2\	COMPSORT_20051119.DOC	1	F	16
	GLB_COALESCE.SAS	1	F	17
NOT_IN_TEST1\	COMPSORT_20051119.DOC	2	F	14
	GLB_COALESCE.SAS	2	F	15
NOT_IN_TEST2\	COMPSORT_20051119.DOC	1	F	18
	GLB_COALESCE.SAS	1	F	19

PRINT: DSN=COMPARE (), NOBS=5, OBS=MAX
WHERE=COMPARE NE 'SAME'
COMPARE: These corresponding files have either DIFF=differences or else are SAME
ROOTPATH1=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST1\
ROOTPATH2=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST2\
compare=DIFF

subroot	flname	nbytes1	nroot1	nbytes2	nroot2	dfflag
BOTH\	BOTH_DIFF.DOC	20480	1	20480	2	F
	BOTH_DIFF.TXT	46	1	15	2	F
BOTH\DIFF\	CPORT.CPT	1920	1	1968	2	F
	CPORT.LOG	1437	1	1475	2	F
	CPORT.SAS	119	1	119	2	F

…

PRINT: DSN=SUMMARY (), NOBS=2, OBS=MAX
Summary Counts
ROOTPATH1=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST1\
ROOTPATH2=C:\Documents and Settings\barnetta\My Documents\FTPCache\PharmaSUG\2006\DIFFTREE\EXAMPLE\TEST2\

DFFLAG	N1	N2	NTOTAL	MISSING1	MISSING2	DIFF1	DIFF2	SAME1	SAME2	COMPTOTAL	NBYTES1	NBYTES2	PASSFAIL
D	7	5	12	3	1	.	.	4	4	12	32	32	FAIL
F	12	10	22	4	2	5	5	3	3	22	44546	44601	FAIL
	19	15	34	7	3	5	5	7	7	34	44578	44633	

Appendix C: TRAVERSE2.SAS macro

```
/*
*************************************************************************
*       PROGRAM NAME:      TRAVERSE2.SAS
*        SAS VERSION:      V8.2
*            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:
*             %TRAVERSE2( rootpath, <, nroot= > <, stop_abort => <, case=> <, debug = >
*                          <, namelength = > )
*             where:
*                    ROOTPATH     - the path to the root of the subtree,
*                    NROOT        = flag to suffix files/data sets with in case calling
*                                 %TRAVERSE2 multiple times
*                    STOP_ABORT   = whether to STOP or ABORT (default is STOP)
*                    DEBUG        = 0=no debug (default), 1=debug
*                    NAMELENGTH   = max filename length (default = 2000)
*                    LIST         = N=no output (default), Y=output
*
*         INPUT FILES:   all files at and below the ROOTPATH
*       OUTPUT FILES:   listing of filenames
*
*MACROS USED/CREATED:      %GLB_WHEREAMI, %GLB_PRTEMPTY, %GLB_EQUIJOIN
*              AUTHOR:   Andy Barnett
*       DATE CREATED:   18OCT2005
*
*   MODIFICATION LOG:
*************************************************************************
* DATE        INITIALS     MODIFICATION DESCRIPTION
*************************************************************************
* MM/DD/YYYY               Complete description of modification made
*                          including reference to source of change.
*
*************************************************************************
*  © 2005-2006 Pharmaceutical Product Development, Inc.
*  All Rights Reserved.
*************************************************************************;
*/
%macro traverse2( rootpath, nroot =, stop_abort = STOP, case = U, debug = 0,
                  namelength = 2000, LIST = N );
      %local id err kerr;
      %let id = &sysmacroname;
      %let err = E%str(RROR:) &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);
      %let list = %qupcase(&list);

      %if %index(.WIN.VMS_AXP.,.&sysscp..) = 0 %then %do;
```

```
              %put &err: Un%str(known) operating system (SYSSCP=&sysscp) - aborting;
              %let kerr = 1;
       %end;
       %if &kerr ne 0 %then %goto err;

%*Check that rootpath exists and is accessible;
       data _null_;
              length rootpath sysmsg $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:          sysmsg = sysmsg();
              put 'E' "RROR: &id: Cannot find/access " rootpath= rc= sysmsg=;
              call symput( 'kerr', '1' );
              %if %qupcase(&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;

       %if &sysscp = WIN %then %do;
%*Add trailing \ if not already present;
              %if %qsubstr(&rootpath,%length(&rootpath),1) ne \
                     %then %let rootpath = &rootpath\;
       %end;
       %else %if &sysscp = VMS_AXP %then %do;
              %if %qsubstr(&rootpath,%length(&rootpath),1) ne ] %then %do;
                     %put &err: No directory specified: ROOTPATH=&rootpath;
                     %let kerr = 1;
              %end;
       %end;
       %if &kerr ne 0 %then %goto err;

%*Find name of lowest level in ROOTPATH;
       data _null_;
              length rootpath $&namelength;
              rootpath = symget( 'rootpath' );
              %if &sysscp = WIN %then %do;
                     rootdir = left(reverse(scan(left(reverse(rootpath)),1,'\')));
                     if substr(rootdir,length(rootdir),1) = ':' then rootdir = '\';
              %end;
              %else %do;
                     rootdir = left( reverse( scan(
                                  left( reverse( scan( rootpath, 2, '[]' ) ) ),
                                                       1, '.' ) ) );
```

```
              %end;
              call symput( 'rootdir', trim( rootdir ) );
              stop;
       run;

%*Get a list of all filenames below each ROOTPATH.
Remember to capture all file versions under OpenVMS,
and to check for System/Hidden files under Windows.
For OpenVMS, we will not use PIPE, because it is slow,
and instead spool to a WORK file;
       %if &sysscp = WIN %then %do;
              %local workpath;
              %let workpath = %sysfunc(pathname(work))\;
              x "dir/s/b/a-d  ""&rootpath"" >""&workpath.traverse_dir&nroot..tmp""";
              x "dir/s/b/as-d ""&rootpath"" >""&workpath.traverse_dir&nroot.s.tmp""";
              x "dir/s/b/ah-d ""&rootpath"" >""&workpath.traverse_dir&nroot.h.tmp""";
              x "dir/s/b/ad   ""&rootpath"" >""&workpath.traverse_dir&nroot.d.tmp""";
              x "dir/s/b/asd  ""&rootpath"" >""&workpath.traverse_dir&nroot.ds.tmp""";
              x "dir/s/b/ahd  ""&rootpath"" >""&workpath.traverse_dir&nroot.dh.tmp""";
              %let tmp = dir/b ""&workpath.traverse_dir&nroot.*.tmp"";
              %let tmp = &tmp >""&workpath.traverse_tmp&nroot..tmp"";
              x "&tmp";
              filename lst "&workpath.traverse_tmp&nroot..tmp";
       %end;
       %else %do;
%*Get all the directories and regular filenames;
              %let tmp = dir/nohead/notrail/nodate/nosize;
              %let tmp = &tmp %scan(&rootpath,1,])...]*.*%str(;)*;
              %let tmp = &tmp/out=sas$worklib:traverse&nroot..tmp;
              x "&tmp";
              %let tmp = dir/nohead/notrail/nodate/nosize;
              %let tmp = &tmp sas$worklib:traverse&nroot..tmp;
              %let tmp = &tmp/out=sas$worklib:traverse&nroot._tmp&nroot..tmp;
              x "&tmp";
              filename lst "sas$worklib:traverse&nroot._tmp&nroot..tmp";
       %end;

%*Now, get the list of file and directory names as a data set;
       data tmp (keep = nroot rootpath subroot flname dfflag lflname compress = yes );

              length rootpath subroot flname infilename $&namelength dfflag $1;
              retain nroot "&nroot" rootpath;
              if _n_ = 1 then do;
                     rootpath = symget( 'rootpath' );
                     if rootpath ne symget( 'rootpath' ) then do;
                            put "E" "RROR: &id: NAMELENGTH=&namelength too small for "
                                   rootpath=;
                            call symput( 'kerr', '1' );
                            %if %qupcase(&stop_abort) = STOP %then stop;
                            %else abort abend;
                            ;
                     end;
%*Create the root directory entry;
                     %if &sysscp = WIN %then %do;
                            subroot = '.\';
                            flname = '.\';
                     %end;
                     %else %do;
                            subroot = '000000]';
```

```
                        flname = "000000.DIR";
                  %end;
                  dfflag = 'D';
                  link output;
            end;
%*Get each file/directory name;
            infile lst pad truncover end = eof;
            input infilename $char&namelength..;
            if infilename ne ' ' then link dir;
            if eof then do;
                  put
                   "NOTE: &id: #(rcds read)        = " n
                  /"NOTE: &id: #(skipped rcds)     = " skip
                  /"NOTE: &id: MAX(LFLNAME)        = " maxlflname
                  ;
                  %if &sysscp ne WIN %then %do;
                  put
                   "NOTE: &id: #(skipped err rcds) = " err;
                  if err then call symput( 'kerr', '1' );
                  %end;
            end;
            return;

dir:        %if &sysscp = WIN %then infilename = "&workpath" || infilename%str(;);
            %if &debug %then put infilename= eof2=%str(;);
            infile dummy filevar = infilename end = eof2 pad truncover;
            do while ( not eof2 );
                  input flname $char&namelength..;
                  n + 1;
                  %if &debug %then put n= flname=%str(;);
                  lflname = length( flname );
                  maxlflname = max( maxlflname, lflname );
                  retain maxlflname 0;
                  %if &sysscp = WIN %then %do;
                        if flname notin ( 'File Not Found' ' ' ) then do;
                              length tmp $&namelength;
                              tmp = left( reverse( flname ) );
                              i = index( tmp, '\' );
                              subroot = substr( left( reverse(
                                    substr( tmp, i ) ) ),
                                          %eval(%length(&rootpath)+1) );
                              if subroot = ' ' then subroot = '.\';
                              flname = left( reverse( substr( tmp, 1, i-1 ) ) );
                              tmp = left(reverse( scan(left(reverse(infilename)),
                                                2,'.\')));
                              if index(upcase(tmp), "%upcase(DIR&nroot.D)") > 0
                              then dfflag = 'D';
                              else dfflag = 'F';
                              link output;
                        end;
                        else if flname = ' ' then link warn;
                  %end;
                  %else %do;
                        if flname ne ' ' then do;
                              if indexc( flname, '%|' ) = 0 then do;
                                    subroot = substr(scan(flname,1,']') || '] ',
                                                %eval(%length(&rootpath)+1) );
                                    if subroot = " " then subroot = "000000]";
                                    flname = scan( flname, 2, ']' );
```

```
                                        if scan( scan(flname,2,'.'), 1, ';' ) = 'DIR'
                                        then dfflag = 'D';
                                        else dfflag = 'F';
                                        link output;
                                  end;
                                  else link err;
                            end;
                            else link warn;
                      %end;
               end;
               eof2 = 0;
               return;

output:        %if &debug %then put n= subroot= flname= dfflag=%str(;);
               output;
               return;

warn:          put 'W' 'ARNING: Skipping record: ' n= flname=;
               skip + 1;
               return;
               %if &sysscp ne WIN %then %do;
err:           put 'E' 'RROR: Cannot recover: ' n= flname=;
               err + 1;
               return;
               %end;
       run;

       filename lst clear;

       %if &debug %then %do;
              %glb_prtempty( tmp,
                     reportcol = subroot flname nroot dfflag,
                     reportwid = 60      60     8     1,
                     reportflow = subroot flname,
                     reportorder = subroot flname nroot )
       %end;
       %if &kerr ne 0 %then %goto err;

       %if &list = Y %then %do;
              proc summary nway missing data = tmp;
                     var lflname;
                     output out = summary&nroot
                            min = min max = max mean = mean n = n nmiss = nmiss;
              run;
              %glb_prtempty( summary&nroot,
                     titles = "NOTE: &id: SUMMARY&nroot, Summary of LENGTH(FLNAME)" )
       %end;

       proc sql;
%*Get the list of unique names;
              create table dirfilenames&nroot as
              select *, monotonic() as n
              from   ( select distinct * from tmp )
              order  by subroot, flname;
       quit;

       %if &list ne N %then %do;
%*Save filenames in a file;
              %let CSV = &herepath.traverse2&nroot._filelist.txt;
```

```
            %put CSV=&csv;
            ods csv file = "&csv";
            ods listing close;
            proc print width = min data = dirfilenames&nroot;
                  var subroot flname nroot rootpath dfflag n;
            run;
            ods listing;
            ods csv close;
            run;
            %glb_prtempty( dirfilenames&nroot,
                  reportcol = subroot flname nroot dfflag n,
                  reportwid = 50          50     8     1      12,
                  reportflow = subroot flname,
                  reportorder = subroot flname nroot )
      %end;
      %goto fin;

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

%fin:  %put &id: End;
%mend traverse2;
```

Appendix D: DIFFTREE macro

```
/*
*************************************************************************
*       PROGRAM NAME:      DIFFTREE.SAS
*        SAS VERSION:      V8.2
*            PURPOSE:      Compare all files beneath two directory subtrees to
*                          classify files in either directory as one of {SAME, DIFF,
*                          MISSING} compared to other corresponding directory/file.
*
*        USAGE NOTES:
*            %DIFFTREE( rootpath1, rootpath2 <, stop_abort => <,where => <, case=>
*                                <, debug = > <, namelength = > )
*            where:
*                  ROOTPATH1     - the path to the root of the 1st subtree,
*                  ROOTPATH2     - the path to the root of the 2nd subtree,
*                  STOP_ABORT    = whether to STOP or ABORT in Windows session
*                                (default is STOP)
*                  WHERE         = a condition, (default is COMPARE NE 'SAME', to skip
*                                printing the list of files that are the same)
*                  CASE          = U -> upcase the path/filenames (default), anything
*                                else case-sensitive
*                  DEBUG         = 0=no debug (default), 1=debug
*                  NAMELENGTH    = max filename length (default = 2000)
*                  LIST          = Y=output (default), N=no output
*
*        INPUT FILES:   all files at and below the two ROOTPATHs
*       OUTPUT FILES:   listing of classified filenames
*
*MACROS USED/CREATED:      %GLB_WHEREAMI, %GLB_PRTEMPTY, %GLB_COALESCE, %GLB_EQUIJOIN
*             AUTHOR:    Andy Barnett
*       DATE CREATED:    18OCT2005
```

```
*
*   MODIFICATION LOG:
*************************************************************************
* DATE       INITIALS     MODIFICATION DESCRIPTION
*************************************************************************
* MM/DD/YYYY              Complete description of modification made
*                         including reference to source of change.
*
*************************************************************************
*  © 2005-2006 Pharmaceutical Product Development, Inc.
*  All Rights Reserved.
*************************************************************************;
*/
%macro difftree( rootpath1, rootpath2,
       stop_abort = STOP, where = COMPARE NE 'SAME', case = U, debug = 0,
       namelength = 2000, list = Y );
       %local id err kerr;
       %let id = &sysmacroname;
       %let err = E%str(RROR:) &id;
       %let kerr = 0;

       %put &id: Begin;
       %put ROOTPATH1=&ROOTPATH1;
       %put ROOTPATH2=&ROOTPATH2;
       %put WHERE=&where;
       %put STOP_ABORT=&STOP_ABORT, CASE=&case, DEBUG=&debug,;
       %put NAMELENGTH=&namelength, LIST=&list;

/*
Set PATHCHARS, FHOSTOPTS, SASAUTOS - depending on platform (WIN or OpenVMS) - where:
       PATHCHARS = :\ for WIN, :[] for VMS
       FHOSTOPTS = null for WIN, FAC=GET SHR=GET for VMS (to allow readonly-access to
              files)
       SASAUTOS = point to default directory (.\ on WIN, [] on VMS)
Also, if WIN, check if .\ is under X:\WIN*, if so stop.
*/
       %local pathchars fhostopts pgmid herepath workpath;
       options ls = 132 ps = 60 noovp compress = yes spool noerrorabend nofmterr
              nodate;
       %let workpath = %sysfunc(pathname(work));
       %if &sysscp = WIN %then %do;
              %let fhostopts = ;
              %let pathchars = ':\';
              %let herepath = .\;
              %let workpath = &workpath\;
              options sasautos = ( ".\", sasautos ) nofullstimer;
       %end;
       %else %if &sysscp = VMS_AXP %then %do;
              %let fhostopts = ,, 'fac = get shr = get';
              %let pathchars = ':[]';
              %let herepath = [];
              options sasautos = ( "[]", sasautos ) cc = cr fullstimer;
       %end;
       %else %do;
              %put &err: Un%str(known) operating system (SYSSCP=&sysscp) - aborting;
              %let kerr = 1;
       %end;
       %if &kerr ne 0 %then %goto err;
```

```
        libname herepath "&herepath" access = readonly;
        %let herepath = %sysfunc(pathname(herepath));
        libname herepath clear;
        %if &sysscp = WIN and %qsubstr(&herepath,%length(&herepath),1) ne \
        %then %let herepath = &herepath\;

        libname rpath1 "&rootpath1" access = readonly;
        libname rpath2 "&rootpath2" access = readonly;
        %let rootpath1 = %sysfunc(pathname(rpath1));
        %let rootpath2 = %sysfunc(pathname(rpath2));
        libname rpath1 clear;
        libname rpath2 clear;
        %if &sysscp = WIN %then %do;
               %if %qsubstr(&rootpath1,%length(&rootpath1),1) ne \
                      %then %let rootpath1 = &rootpath1\;
               %if %qsubstr(&rootpath2,%length(&rootpath2),1) ne \
                      %then %let rootpath2 = &rootpath2\;
        %end;
        %else %do;
               %if %qsubstr(&rootpath1,%length(&rootpath1),1) ne ] %then %let kerr = 1;
               %else %if %qsubstr(&rootpath2,%length(&rootpath2),1) ne ]
                      %then %let kerr = 1;
               %if &kerr ne 0
                      %then %put &err: ROOTPATH1/ROOTPATH2 does not look like directory;
        %end;

        %put SYSSCP=&sysscp, HEREPATH=&herepath,;
        %put FHOSTOPTS=&fhostopts, PATHCHARS=&pathchars,;
        %put ROOTPATH1=&rootpath1;
        %put ROOTPATH2=&rootpath2;
        %if %index(%upcase(&herepath),:\WIN) > 0 %then %do;
               %put &err: Current Directory may be system area: HEREPATH=&herepath;
               %let kerr = 1;
        %end;
        %if &kerr ne 0 %then %goto err;

%*Get a list of all filenames below each ROOTPATH.
Remember to capture all file versions under OpenVMS,
and to check for System/Hidden files under Windows;
        %traverse2( &rootpath1, nroot = 1, list = &list, debug = &debug )
        %traverse2( &rootpath2, nroot = 2, list = &list, debug = &debug )

        proc sql;
               create table dirfilenames as
               select subroot, flname, dfflag, n, nroot, rootpath
               %if %qupcase(&case) = U %then %do;
               from   ( select     upcase( subroot ) as subroot,
                                   upcase( flname ) as flname,
                                   dfflag, n, nroot, rootpath
                             from   dirfilenames1 )
                      union all
                      ( select     upcase( subroot ) as subroot,
                                   upcase( flname ) as flname,
                                   dfflag, n, nroot, rootpath
                             from   dirfilenames2 )
               %end;
               %else %do;
               from   ( select     subroot, flname, dfflag, n, nroot, rootpath
                             from   dirfilenames1 )
```

```
                  union all
                  ( select     subroot, flname, dfflag, n, nroot, rootpath
                        from   dirfilenames2 )
            %end;
            order  by subroot, flname, dfflag, nroot;
      quit;

      %if %qupcase(&list) ne N %then %do;
%*Save filenames in a file;
            ods csv file = "&herepath.difftree_filelist.txt";
            ods listing close;
            proc print width = min data = dirfilenames;
                  var subroot flname dfflag n nroot rootpath;
            run;
            ods listing;
            ods csv close;
            run;
            %glb_prtempty( dirfilenames,
                  titles = "ROOTPATH1=&rootpath1" @ "ROOTPATH2=&rootpath2", dlm = @,
                  reportcol = subroot flname dfflag n  nroot,
                  reportwid = 50          50     6      12 5,
                  reportflow = subroot flname,
                  reportorder = subroot,
                  reportskip = subroot )
      %end;

%*Now find DIFF, SAME, MISSING tables;
      proc sql;
            create table missing as
            select %glb_coalesce( subroot flname dfflag n nroot )
            from   dirfilenames ( keep = subroot flname nroot dfflag n
                        where = ( nroot = '1' ) ) a
                  full join
                  dirfilenames ( keep = subroot flname nroot dfflag n
                        where = ( nroot = '2' ) ) b
            on     %glb_equijoin( subroot flname dfflag )
            where  a.nroot is null or b.nroot is null
            order  by subroot, flname, dfflag, nroot
            ;

            create table match as
            select a.subroot, a.flname, a.dfflag,
                  a.nroot as nroot1, b.nroot as nroot2,
                  a.rootpath as rootpath1, b.rootpath as rootpath2
            from   dirfilenames ( where = ( nroot = "1" ) ) a
                  inner join
                  dirfilenames ( where = ( nroot = "2" ) ) b
            on     %glb_equijoin( subroot flname dfflag )
            order  by subroot, flname, dfflag
            ;
      quit;

      data compare ( keep = subroot flname dfflag nroot1 nroot2 compare
                        nbytes1 nbytes2 );
            set match end = eof;
            length compare $4 rcd1 rcd2 $32767 filename1 filename2 $&namelength
                  sysmsg $2000 lastsubroot lastrootpath1 lastrootpath2 $1;
%*RETAIN on RCD1/2 needed to prevent un.initialized message, even though FGET assigns
a value;
```

```
            retain rcd1 rcd2 ' ';
            filename1 = flname;
            filename2 = flname;
            lastsubroot = substr( subroot, length( subroot ), 1 );
            if rootpath1 = ' ' or rootpath2 = ' ' then do;
                  put "E" "RROR: ROOTPATH1/ROOTPATH2 is null"; link abort;
            end;
            lastrootpath1 = substr( rootpath1, length( rootpath1 ), 1 );
            lastrootpath2 = substr( rootpath2, length( rootpath2 ), 1 );
            %if &sysscp = WIN %then %do;
                  if subroot ne ' ' then do;
                        if index( subroot, '\' ) > 0 and lastsubroot ne '\'
                        then do;
                              filename1 = trim( subroot ) || '\' || filename1;
                              filename2 = trim( subroot ) || '\' || filename2;
                        end;
                        else do;
                              filename1 = trim( subroot ) || filename1;
                              filename2 = trim( subroot ) || filename2;
                        end;
                  end;
                  if index( rootpath1, '\' ) > 0 and lastrootpath1 ne '\'
                        then filename1 = '\' || filename1;
                  if index( rootpath2, '\' ) > 0 and lastrootpath2 ne '\'
                        then filename2 = '\' || filename2;
                  if rootpath1 ne ' ' then filename1 = trim( rootpath1 )
                        || filename1;
                  if rootpath2 ne ' ' then filename2 = trim( rootpath2 )
                        || filename2;
            %end;
            %else %do;
                  if subroot ne ' ' then do;
                        if lastsubroot = '.' then do;
                              filename1 = substr(subroot,1,length(subroot)-1)
                                    || ']' || filename1;
                              filename2 = substr(subroot,1,length(subroot)-1)
                                    || ']' || filename2;
                        end;
                        else if lastsubroot = ']' then do;
                              filename1 = trim( subroot ) || filename1;
                              filename2 = trim( subroot ) || filename2;
                        end;
                        else do; put "E" "RROR: " lastsubroot=; link abort; end;
                  end;
%*Now, have subroot]filename, or filename;
                  if lastrootpath1 in ( ']' '.' ) then do;
                        if index( filename1, ']' ) > 0 then do;
                              if substr( filename1, 1, 1 ) = '.' then filename1 =
                                    substr(rootpath1,1,length(rootpath1)-1)
                                          || filename1;
                              else if substr( filename1, 1, 1 ) = ']'
                              then filename1 =
                                    substr(rootpath1,1,length(rootpath1)-1)
                                          || filename1;
                              else filename1 =
                                    substr(rootpath1,1,length(rootpath1)-1)
                                          || '.' || filename1;
                        end;
                        else filename1 = substr(rootpath1,1,length(rootpath1)-1)
```

```
                        || ']' || filename1;
                end;
                else do; put "E" "RROR: " lastrootpath1=; link abort; end;
                if lastrootpath2 in ( ']' '.' ) then do;
                        if index( filename2, ']' ) > 0 then do;
                                if substr( filename2, 1, 1 ) = '.'
                                then filename2 =
                                        substr(rootpath2,1,length(rootpath2)-1)
                                                || filename2;
                                else if substr( filename2, 1, 1 ) = ']'
                                then filename2 =
                                        substr(rootpath2,1,length(rootpath2)-1)
                                                || filename2;
                                else filename2 =
                                        substr(rootpath2,1,length(rootpath2)-1)
                                                || '.' || filename2;
                        end;
                        else filename2 = substr(rootpath2,1,length(rootpath2)-1)
                                || ']' || filename2;
                end;
                else do; put "E" "RROR: " lastrootpath2=; link abort; end;
                filename1 = tranwrd( filename1, ".000000]", "]" );
                filename2 = tranwrd( filename2, ".000000]", "]" );
        %end;

        filename1 = left( filename1 );
        filename2 = left( filename2 );
        compare = 'SAME';

        if dfflag = 'F' then link compare;
        output;
        return;

compare:  rc1 = filename( 'fref1', trim( filename1 ) &fhostopts );
        if rc1 ne 0 then link abort;
        rc2 = filename( 'fref2', trim( filename2 ) &fhostopts );
        if rc2 ne 0 then link abort;
        fid1 = fopen( 'fref1', 'I', 32767, 'B' );
        if fid1 <= 0 then link abort;
        fid2 = fopen( 'fref2', 'I', 32767, 'B' );
        if fid2 <= 0 then link abort;
        eof1 = fread( fid1 );
        eof2 = fread( fid2 );
        nbytes1 = 0;
        nbytes2 = 0;
        do while ( compare = 'SAME' and not ( eof1 or eof2 ) );
                get1 = fget( fid1, rcd1, 32767 );
                get2 = fget( fid2, rcd2, 32767 );
                nbytes1 + frlen( fid1 );
                nbytes2 + frlen( fid2 );
                if not ( get1 = 0 and get2 = 0 ) then compare = 'DIFF';
                else if nbytes1 ne nbytes2 then compare = 'DIFF';
                else if rcd1 ne rcd2 then compare = 'DIFF';
                eof1 = fread( fid1 );
                eof2 = fread( fid2 );
        end;
        if not ( eof1 and eof2 ) then compare = 'DIFF';
        rc = fclose( fid1 );
        rc = fclose( fid2 );
```

```
              rc = filename( 'fref1' );
              rc = filename( 'fref2' );
              return;

abort:        sysmsg = sysmsg();
              put 'E' "RROR: " sysmsg= / _all_;
              call symput( 'kerr', '1' );
              stop;

       run;
       %glb_prtempty( missing, dlm = @, printnull = N,
              titles = "MISSING: These files were found without corresponding files "
                           "in other ROOTPATH"
                     @ "ROOTPATH1=&rootpath1"
                     @ "ROOTPATH2=&rootpath2",
              reportcol = subroot flname nroot dfflag n,
              reportwid = 50      50     5     6      12,
              reportflow = subroot flname,
              reportorder = subroot flname,
              reportskip = subroot )
       %glb_prtempty( compare, where = &where, printnull = N, dlm = @,
              titles = "COMPARE: These corresponding files have either "
                           "DIFF=differences or else are SAME"
                     @"ROOTPATH1=&ROOTPATH1"
                     @"ROOTPATH2=&ROOTPATH2",
              reportby = compare,
              reportcol = subroot flname nbytes1 nroot1 nbytes2 nroot2 dfflag,
              reportwid = 40      40     14      6      14      6      6,
              reportflow = subroot flname,
              reportorder = subroot,
              reportskip = subroot )

%*Show summary counts;
       data summary_levels ( keep = DFFLAG NROOT TYPE COUNT );
              length DFFLAG NROOT $1 TYPE $7;
              retain COUNT 0;
              do DFFLAG = 'D', 'F';
                     do NROOT = '1', '2';
                            do TYPE = 'DIR', 'MISSING', 'DIFF', 'SAME';
                                   output;
                            end;
                     end;
              end;
              stop;
       run;

       proc sql;
              create table summary as
              select DFFLAG, NROOT, TYPE, sum( NBYTES ) as nbytes, count(*) as COUNT
              from   ( select     DFFLAG, NROOT, NBYTES, TYPE
                           from   ( select     DFFLAG, NROOT, . AS NBYTES,
                                               'DIR' as TYPE
                                         from   dirfilenames )
                                  union all
                                  ( select     DFFLAG, NROOT, . AS NBYTES,
                                               'MISSING' as TYPE
                                         from   missing )
                                  union all
                                  ( select     DFFLAG, NROOT1 as NROOT,
```

```
                                        NBYTES1 as NBYTES, COMPARE as TYPE
                                from    compare )
                        union all
                        ( select        DFFLAG, NROOT2 as NROOT,
                                        NBYTES2 as NBYTES, COMPARE as TYPE
                                from    compare )
                )
        group  by DFFLAG, NROOT, TYPE;

        create table summary as
        select %glb_coalesce( DFFLAG NROOT TYPE COUNT ), a.NBYTES
        from   summary a right join summary_levels b
        on     %glb_equijoin( DFFLAG NROOT TYPE )
        order  by DFFLAG, NROOT, TYPE;
quit;

data summary ( keep = dfflag n1 n2 ntotal nbytes1 nbytes2
                      missing1 missing2 diff1 diff2 same1 same2
                      comptotal passfail );
        merge  summary ( where = ( nroot = '1' )
                         rename = ( NBYTES = B1 COUNT = K1 ) )
               summary ( where = ( nroot = '2' )
                         rename = ( NBYTES = B2 COUNT = K2 ) )
               ;
        by dfflag type;
        if not ( first.type and last.type ) then abort abend;
        if type = 'MISSING' then do; MISSING1 = K1; MISSING2 = K2; end;
        else if type = 'DIFF' then do; DIFF1 = K1; DIFF2 = K2; end;
        else if type = 'SAME' then do; SAME1 = K1; SAME2 = K2; end;
        else if type = 'DIR' then do; N1 = K1; N2 = K2; end;
        else do; put "E" "RROR: Oops!"; abort abend; end;
        retain N1 N2 NBYTES1 NBYTES2 0 MISSING1 MISSING2 DIFF1 DIFF2 SAME1 SAME2;
        if B1 > 0 then NBYTES1 + B1;
        if B2 > 0 then NBYTES2 + B2;
        if last.dfflag;
        NTOTAL = sum( N1, N2 );
        COMPTOTAL = sum( MISSING1, MISSING2, DIFF1, DIFF2, SAME1, SAME2 );
        if sum( MISSING1, MISSING2, DIFF1, DIFF2 ) > 0 or NBYTES1 ne NBYTES2
               then PASSFAIL = 'FAIL';
        else if sum( N1, N2 ) = sum( SAME1, SAME2 ) then PASSFAIL = 'PASS';
        else PASSFAIL = 'FAIL';
run;
%glb_prtempty( SUMMARY, dlm = @,
        titles = "Summary Counts" @ "ROOTPATH1=&rootpath1"
               @ "ROOTPATH2=&rootpath2",
        prstmts = %quote(
               by dfflag;
               id dfflag;
               var N1 N2 NTOTAL MISSING1 MISSING2 DIFF1 DIFF2 SAME1 SAME2
                      COMPTOTAL NBYTES1 NBYTES2 PASSFAIL;
               sum N1 N2 NTOTAL MISSING1 MISSING2 DIFF1 DIFF2 SAME1 SAME2
                      COMPTOTAL NBYTES1 NBYTES2;
        )
)
%goto fin;

%err:  %put &err: Aborting...;
       %put &err: ... ROOTPATH1=&rootpath1,;
       %put &err: ... ROOTPATH2=&rootpath2,;
```

```
	%put &err: ... STOP_ABORT=&STOP_ABORT, WHERE=&where, CASE=&case, DEBUG=&debug,;
	%put &err: ... NAMELENGTH=&namelength;
	%if %qupcase(&stop_abort) = STOP %then %goto fin;
	e%str(ndsas;)

%fin:  %put &id: End;
%mend difftree;
```

Appendix E: MACROS.SAS

```
%macro init;
/*
Set PATHCHARS, FHOSTOPTS, SASAUTOS, PGMID, WORKPATH - depending on platform (WIN or
OpenVMS) - where:
	PATHCHARS = :\ for WIN, :[] for VMS
	FHOSTOPTS = null for WIN, FAC=GET SHR=GET for VMS (to allow readonly-access to
		files)
	SASAUTOS = point to default directory (.\ on WIN, [] on VMS)
	PGMID = name of this program
	WORKPATH = path to WORK library directory
Also, if WIN, check if .\ is under X:\WIN*, if so stop.
*/
	%global pathchars fhostopts pgmid herepath workpath;
	options ls = 132 ps = 60 noovp mprint nosymbolgen compress = yes spool
		noerrorabend nofmterr nodate;
	%if &sysscp = WIN %then %do;
		%let fhostopts = ;
		%let pathchars = ':\';
		%let herepath = .\;
		options sasautos = ( ".\", sasautos ) nofullstimer noxwait xsync;
		%let workpath = %sysfunc(pathname(work))\;
	%end;
	%else %if &sysscp = VMS_AXP %then %do;
		%let fhostopts = ,, 'fac = get shr = get';
		%let pathchars = ':[]';
		%let herepath = [];
		options sasautos = ( "[]", sasautos ) cc = cr fullstimer xwait;
		%let workpath = %sysfunc(pathname(work));
	%end;
	%else %do;
		%let tmp = Un%str(known) operating system (SYSSCP=&sysscp) - aborting;
		%put E%str(RROR): &tmp;
		e%str(ndsas);
	%end;
	%let pgmid = &sysprocessname;

	libname herepath "&herepath" access = readonly;
	%let herepath = %sysfunc(pathname(herepath));
	libname herepath clear;
	%if &sysscp = WIN and %qsubstr(&herepath,%length(&herepath),1) ne \
	%then %let herepath = &herepath\;

	%put SYSSCP=&sysscp, HEREPATH=&herepath, WORKPATH=&workpath,;
	%put FHOSTOPTS=&fhostopts, PATHCHARS=&pathchars,;
	%put PGMID=&pgmid;

	%if %index(%upcase(&herepath),:\WIN) > 0 %then %do;
		%let tmp = Current Directory may be system area: HEREPATH=&herepath;
		%put E%str(RROR): &tmp;
```

```
            %put E%str(RROR): ... aborting;
            e%str(ndsas);
      %end;

      %global ls lsfmt border title1 title2 ntitle
            foot1 foot2 foot3 foot4 foot5 nfoot;
      %local i;
      %let ls = %sysfunc(getoption(linesize));
      %let lsfmt = $&ls..;
      %let border = %sysfunc(repeat(-,&ls-1));
      %let title1 = "PPD Inc.";
      %let title2 = "&border";
      %let ntitle = 2;
      %let foot1 = "%sysfunc(putc(&border,&lsfmt))";
      %let foot2 = %sysfunc(quote(&pgmid));
      %let foot2 = &foot2 "%sysfunc(repeat(%str( ),&ls-%length(&pgmid)-1))";
      %let foot3 = SAS&sysver on &sysscpl – RUNDATE: &sysdate9:&systime;
      %let foot3 = "%sysfunc(putc(&foot3,&lsfmt))";
      %let nfoot = 3;

      %do i = 1 %to &ntitle; title&i &&&title&i; %end;
      %do i = 1 %to &nfoot; footnote&i &&&foot&i; %end;
%mend init;

%macro err( err, stop_abort );
      do;
            if 0 then stop_abort = 'ABORT';
            stop_abort = "%qupcase(&stop_abort)";
            err = &err;
            link err;
      end;
%mend err;

/**$START$*****************************************************************/
/*
/*     PROGRAM:     GLB_COALESCE.SAS
/*   DIRECTORY:
/*      AUTHOR:     Andy Barnett
/*DATE CREATED:     05/27/97
/*     PURPOSE:     Creates sequence of COALESCE() calls for use with PROC SQL
/*       USAGE:     %GLB_COALESCE( vars )
/*            For example, %glb_coalesce( provar invvar patvar ) creates:
/*                  coalesce( a.provar, b.provar ) as provar,
/*                  coalesce( a.invvar, b.invvar ) as invvar,
/*                  coalesce( a.patvar, b.patvar ) as patvar
/*
/*MODIFICATION LOG:
/*  DATE     USERID       DESCRIPTION
/*  ---------------------------------------------------------------------
/*  mm/dd/yy
/*  01/14/02 barnetta     -- added END=Y parameter to tell not to
/*                        add a trailing comma when it is the last
/*                        in a list
/*
/* © 1997-2006, Pharmaceutical Product Development, Inc.
/* All Rights Reserved.
/**$STOP$******************************************************************/
%macro glb_coalesce( vars, end = Y );
      %local n var;
```

```
      %let n = 0;
      %do %while (%qscan(&vars,&n+1,%str( )) ne);
            %let n = %eval(&n+1);
            %let var = %scan(&vars,&n,%str( ));
            %if &n > 1 %then ,;
            coalesce( a.&var, b.&var ) as &var
      %end;
      %if not ( %qupcase(&end) = Y or %bquote(&vars) = %str() ) %then ,;
%mend glb_coalesce;

/**$START$************************************************************/
/*
/*     PROGRAM:     GLB_COMMA.SAS
/*     VERSION:     V1B-20020114
/*   DIRECTORY:
/*      AUTHOR:     Andy Barnett
/*DATE CREATED:     05/27/97
/*     PURPOSE:     Creates comma-separated list of words (can optionally use
/*                  another delimiter, or define another delimiter for the words)
/*       USAGE:     %GLB_COMMA( words <,dlm = %str(,)> <,strdlm = %str( )> )
/*            For example, %glb_comma( provar invvar patvar ) creates:
/*                  provar, invvar, patvar
/*
/*MODIFICATION LOG:
/*  DATE     USERID       DESCRIPTION
/*  ---------------------------------------------------------------
/*  mm/dd/yy
/*  01/14/02 barnetta     -- added END=Y parameter to tell %COMMA not to
/*                        add a trailing delimiter when it is the last
/*                        in a list
/*
/* © 1997-2006, Pharmaceutical Product Development, Inc.
/* All Rights Reserved.
/**$STOP$*************************************************************/
%macro glb_comma( words, dlm = %str(,), strdlm = %str( ), end = Y );
      %local n;
      %let n = 0;
      %do %while (%qscan(&words,&n+1,&strdlm) ne);
            %let n = %eval(&n+1);
            %if &n > 1 %then %unquote(&dlm);
            %scan(&words,&n,&strdlm)
      %end;
      %if not ( %qupcase(&end) = Y or %bquote(&words) = %str() ) %then
%unquote(&dlm);
%mend glb_comma;

/**$START$************************************************************/
/*     PROGRAM:     GLB_EQUIJOIN.SAS
/*   DIRECTORY:
/*      AUTHOR:     Andy Barnett
/*DATE CREATED:     05/27/97
/*     PURPOSE:     Creates equijoin clause for use with SELECT...*JOIN...ON;
/*       USAGE:     %GLB_EQUIJOIN( a, b = ) where #a = #b, and if B is not specified,
/*                  it defaults to whatever value A is given.
/*            For example,
/*                  %glb_equijoin( provar invvar patvar ) and
/*                  %glb_equijoin( provar invvar patvar, b = provar invvar patvar )
/*            are equivalent and create:
/*                  a.provar = b.provar and a.invvar = b.invvar and
```

```
/*                a.patvar = b.patvar
/*           On some large joins, it is useful to create indexes on the
/*           equijoin variables for each data set (prior to the join):
/*                create index keylis on a ( %glb_comma( provar invvar patvar ) );
/*                create index keylis on b ( %glb_comma( provar invvar patvar ) );
/*
/*MODIFICATION LOG:
/*  DATE     USERID       DESCRIPTION
/*  ---------------------------------------------------------------
/*  970618   BARNETTA     changed second parameter to B= (keyword)
/*  020114   barnetta     -- added END=Y parameter to tell not to
/*                        add a trailing comma when it is the last
/*                        in a list
/* © 1997-2006, Pharmaceutical Product Development, Inc.
/* All Rights Reserved.
/**$STOP$*************************************************************/
%macro glb_equijoin( a, b =, end = Y );
      %if %quote(&b) = %then %let b = &a;
      %local n;
      %let n = 0;
      %do %while (%qscan(&a,&n+1,%str( )) ne);
            %let n = %eval(&n+1);
            %if &n > 1 %then and;
            a.%scan(&a,&n,%str( )) = b.%scan(&b,&n,%str( ))
      %end;
      %if not ( %qupcase(&end) = Y or %bquote(&a) = %str() ) %then ,;
%mend glb_equijoin;

/*
**********************************************************************
*        CLIENT NAME:
*       PROGRAM NAME:   GLB_PRTEMPTY.SAS
*  DIRECTORY/ACCOUNT:
*             AUTHOR:   Andy Barnett
*       DATE CREATED:   10/01/2003
*            PURPOSE:
*        INPUT FILES:
*       OUTPUT FILES:
*        USAGE NOTES:
*              NOTES:
*   MODIFICATION LOG:
**********************************************************************
* DATE          BY              DESCRIPTION
**********************************************************************
* MM/DD/YYYY-USERID Complete description of modification made
*                   including reference to source of change.
* 01/14/2004-BARNETTA     -- added PRINTNULL parm
* 04/02/2004-BARNETTA     -- added REPORTCOL, REPORTWID, REPORTORDER,
*                   REPORTFLOW, REPORTSKIP, REPORTLABEL parms
* 05/11/2004-BARNETTA     -- added parm to allow toggling standard header
*                         STDTITLE=Y/N
*                         -- added title/footer delimiter DLM=\
* 06/24/2004-BARNETTA     -- made all created macro vars local to prevent collisions
*                         with outer macro vars
* 07/06/2004-BARNETTA     -- allowed REPORTSKIP= to specify multiple BREAK vars
* 07/09/2004-BARNETTA     -- fixed problem with PROC SQL not allowing tables with no
*                         columns
* 10/21/2004-BARNETTA     -- fixed occasional problem with sorting data set twice
* 09/23/2005-BARNETTA     -- added REPORTBY= parm
```

```
************************************************************************
*  © 2003-2006, Pharmaceutical Product Development, Inc.
*  All Rights Reserved.
************************************************************************;
*/
%macro glb_prtempty( dsn, stdtitle = Y, titles =, foots =, propts =, obs = MAX,
       where =, prstmts =, nobyline =, sortby =, print = Y, debug = 0, printnull = Y,
       reportby =, reportcol =, reportwid =, reportorder =, reportflow = &reportcol,
       reportskip =, reportlabel =, dlm = \ );

       %global ntitle nfoot;
       %local id title foot i j iby noby prdsn lib mem err warn nt memlabel wtitle
              nf col;
       %let id = &sysmacroname;
       %let err = E%str(RROR: &id);
       %let warn = W%str(ARNING: &id);
       %if &debug = 1 %then %PUT &id:  REPORTING PROCESS MACRO TO SETUP HEADER FOOTER;
       %if %qupcase(&print) ne Y %then %goto fin;
       %let obs = %upcase(&obs);
       %let dsn = %upcase(&dsn);
       %if %index(&dsn,.) = 0 %then %do; %let lib = WORK; %let mem = &dsn; %end;
       %else %do; %let lib = %scan(&dsn,1,.); %let mem = %scan(&dsn,2,.); %end;
%*Adjust titles;
       %if %qupcase(&stdtitle) = Y %then %do;
              %if %bquote(&titles) ne %then %let titles = STANDARD &dlm &titles;
              %else %let titles = STANDARD;
       %end;
%*Make copy of data - this is necessary if any of (1) WHERE specified, (2) SORTBY
specified, or (3) &DSN is a view, is true;
       %if &debug = 1 %then put &id: Copying DSN=&dsn to WORK.GLB_PRTEMPTY;
       %let prdsn = WORK.GLB_PRTEMPTY;
       %if %quote(&sortby) ne %then %do;
              proc sort data = &dsn out = &prdsn;
                     %if %quote(&where) ne %then where &where%str(;);
                     by &sortby;
              run;
       %end;
       %else %do;
              data &prdsn;
                     set &dsn;
                     %if %quote(&where) ne %then where &where%str(;);
              run;
       %end;
%*Count number of obs in &prdsn (copy of &dsn) - cannot count this way if &dsn is
view;
       %local __nobs;
       data _null_;
              if 0 then set &prdsn nobs = __nobs;
              call symput( '__nobs', compress( put( __nobs, ??best. ) ) );
              stop;
       run;

%*Print titles -- first title is default;
       %local nt j;
       %let nt = 0;
       %let j = 0;
       %if %qupcase(&titles) = NONE %then %goto byline;
       %do %while (%qscan(&titles,&j+1,&dlm) ne);
              %let j= %eval(&j+1);
```

```
            %let nt = %eval(&nt+1);
            %let title = %scan(&titles,&j,&dlm);
            %if &ntitle+&nt > 10 %then %goto maxt;
            %if &title = STANDARD %then %do;
                  %let memlabel = ;
                  proc sql noprint;
                        select case
                                    when memlabel is null then memlabel
                                    else quote(trim(memlabel))
                              end as memlabel
                        into   :memlabel
                        from   dictionary.tables
                        where  libname = "&lib" and upcase( memname ) = "&mem"
                        ;
                  quit;
                  %let title = "PRINT: DSN=%upcase(&dsn) (" &memlabel;
                  %let title = &title "), NOBS=&__nobs, OBS=&obs";
                  title%eval(&ntitle+&nt) &title;
                  %if %quote(&where) ne %then %do;
                        %local wtitle;
                        %let wtitle = %sysfunc(quote(&where));
                        %let wtitle = %substr(&wtitle,2,%length(&wtitle)-2);
                        %let nt = %eval(&nt+1);
                        title%eval(&ntitle+&nt) "WHERE=&wtitle";
                  %end;
            %end;
            %else %do;
                  title%eval(&ntitle+&nt) &title;
            %end;
      %end;

%byline:
      %if %quote(&nobyline) ne %then %do;
            options nobyline;
            %let title =;
            %let iby = 0;
            %let j = 0;
            %do %while (%qscan(&nobyline,&iby+1,%str( )) ne);
                  %let iby = %eval(&iby+1);
                  %let noby = %scan(&nobyline,&iby,%str( ));
                  %if %quote(&noby) = %str(/) %then %do;
%*Line break specified;
                        %if %quote(&title) ne %then %do;
                              %let nt = %eval(&nt+1);
                              %if &ntitle+&nt > 10 %then %goto maxt;
                              title%eval(&ntitle+&nt) "&title";
                              %let title =;
                              %let j = 0;
                        %end;
                  %end;
                  %else %do;
                        %let j = %eval(&j+1);
                        %if &j > 1 %then
                              %let title = &title, %upcase(&noby)=#byval(&noby);
                        %else  %let title = %upcase(&noby)=#byval(&noby);
                  %end;
            %end;
            %if %quote(&title) ne %then %do;
                  %let nt = %eval(&nt+1);
```

```
                %if &ntitle+&nt > 10 %then %goto maxt;
                title%eval(&ntitle+&nt) "&title"%str(;);
        %end;
    %end;

%*Setup footers;
    %if %quote(&foots) = %then %let nf = 0;
    %else %if %quote(&foots) ne NONE %then %do;
        %let nf = 0;
        %do %while (%qscan(&foots,&nf+1,&dlm) ne );
            %let nf = %eval(&nf+1);
            %let foot = %scan(&foots,&nf,&dlm);
            %if &nfoot + &nf + 1 > 10 %then %goto maxf;
            footnote%eval(&nfoot+&nf) &foot;
        %end;
    %end;
    %else %let nf = 0;

    %if &__nobs = 0 and %qupcase(&printnull) = Y %then %do;
%*Print special page if data set is empty;
        %let title = "&warn: No observations found in DSN=&dsn";
        %let nt = %eval(&nt+1);
        %if &ntitle + &nt > 10 %then %goto maxt;
        title%eval(&ntitle+&nt) &title;
        data work.glb_prtempty;
            if 0 then set &prdsn;
            output;
            stop;
        run;
        %let prdsn = WORK.GLB_PRTEMPTY;
    %end;
    %if %quote(&reportcol) = %then %do;
        proc print uniform width = min &propts
            data = &prdsn %if %quote(&obs) ne MAX %then (obs=&obs);;
            &prstmts;
            %if %quote(&nobyline) ne %then %do;
                by %glb_compress(&nobyline,dlm=/);
                pageby %glb_last(&nobyline);
            %end;
        run;
    %end;
    %else %do;
        %let reportcol = %upcase(&reportcol);
        %let reportflow = %upcase(&reportflow);
        %let reportlabel = %upcase(&reportlabel);
        %let reportorder = %upcase(&reportorder);
        %if %quote(&reportby) ne %then %do;
            proc sort data = &prdsn;
                by &reportby;
            run;
        %end;
        proc report headline headskip split = ' ' spacing = 1 missing
            data = &prdsn %if %quote(&obs) ne MAX %then (obs=&obs);;
            %if %quote(&reportby) ne %then by &reportby%str(;);
            column &reportcol;
            %let i = 0;
            %do %while (%qscan(&reportcol,&i+1,%str( )) ne);
                %let i = %eval(&i+1);
                %let col = %scan(&reportcol,&i,%str( ));
```

```
                        define &col /
                               %if %index(%quote( &reportorder ),%quote( &col )) > 0
                               %then order;
                               %else display;
                               %if %scan(&reportwid,&i,%str( )) ne @
                                      %then width = %scan(&reportwid,&i,%str( ));
                               %if %index(%quote( &reportflow ),%quote( &col )) > 0
                                      %then flow;
                               %if %index(%quote( &reportlabel ),%quote( &col )) > 0
                                      %then "&col";
                               ;
                  %end;
                  %if %quote(&reportskip) ne %then %do;
                        %let skipi = 0;
                        %do %while ( %qscan(&reportskip,&skipi+1,%str( )) ne );
                               %let skipi = %eval(&skipi+1);
                               %let skipcmd = %scan(&reportskip,&skipi,%str( ));
                               break after %scan(&skipcmd,1,/)
                                      / %if %qscan(&skipcmd,2,/) = %then skip;
                                        %else %scan(&skipcmd,2,/); ;
                        %end;
                  %end;
            run;
      %end;
options byline;
%*Clear titles;
      title%eval(&ntitle+1);
      footnote%eval(&nfoot+1);
%*Delete GLB_PRTEMPTY data set if present;
      %if &prdsn = WORK.GLB_PRTEMPTY %then %do;
            proc datasets nolist library = work;
                  delete glb_prtempty / mt = data;
            quit;
      %end;
      %goto fin;
%maxt: %put &err: TITLE%eval(&ntitle+&nt) is too high for TITLE=&title;
      endsas;
      %goto fin;
%maxf: %put &err: FOOTNOTE%eval(&nfoot+&nf) is too high for FOOT=&foot;
      endsas;
      %goto fin;
%fin:  %if &debug = 1 %then %put &id: End;
%mend glb_prtempty;
```