

A Macro to Produce a SAS® Data Set Containing the List of File Names Found in the Requested Windows or UNIX Directory

Mike Goulding, Experis, Portage, MI, USA

ABSTRACT

Clinical programmers often need to perform a particular process for each file that exists in a specific directory, on Windows or UNIX. For example, consider a directory that contains SAS® V5 transport files, which need to be converted back to standard data sets, perhaps as part of doing a final quality check prior to regulatory submission. To run through the conversion step for all these files dynamically, somehow the programmer must first create a data structure which contains the file names in the target directory.

This paper presents the macro `dir_contents`, which captures all file names from the requested directory, and returns the file names as observations within a SAS® data set. In this structure, the data set can be readily processed by a subsequent macro do-loop, to perform whatever procedure might be appropriate. The macro performs basic error checking, and supports filtering the requested directory by file extension. The macro obtains the information using SAS® software functions rather than system-specific commands, to ensure complete portability between Windows and UNIX.

INTRODUCTION

SAS® software is widely used on two directory-based operating systems: Windows and UNIX. One of the most basic tasks common to both of these systems is to determine which files exist within a particular directory. File lists can be generated via the command line interface, using a system-specific command (`dir` for Windows, `ls` on UNIX), but those commands only display the file list on the screen; the list is not immediately usable by a SAS® program. Numerous methods exist to capture the file list based on using the PIPE option of the FILENAME statement, but implementations are unique to each operating system, and they also involve somewhat esoteric parsing of the command output stream.

The `dir_contents` macro was written with these challenges in mind. The macro provides a straightforward, reusable means of obtaining the directory list and saving it in the familiar form of a SAS® data set. The design concept assumes that in this form, the file list can then be used as metadata to drive any type of downstream user-written macro which iterates for each file name and performs a SAS® process that operates on each file in the directory.

Here are just a few possible applications where the directory information returned by the macro could be leveraged in a subsequent macro %do loop:

- Import all Excel or .csv files found in one directory
- Find the most recently created file in a directory and process only that file
- Scan all log files in a directory for critical errors and warning messages

One illustrative example will be shown in detail, later in the paper. The macro's scope lends itself to a modular approach; details of extracting directory contents are de-coupled from any dependent macro activity that follows.

MACRO KEYWORD PARAMETERS

Parameter	Description
DIR	Full name of directory to process. REQUIRED, no default.
EXT	File extension to apply as a filter. Optional, default is no filtering: all file names are returned.
DSOUT	Output data set name, one- or two- level. Optional, default is work.dir_contents.
ATTRIBS	Flag (Y or N) to pull in additional file attributes such as creation date. Optional, default = N.

Table 1. Macro parameters

ERROR CHECKING

The macro performs basic error checking and will exit with an informative message if any of these conditions is found:

- The DIR required parameter is blank.
- The specified directory does not exist.
- The specified directory could not be opened.
- The directory has zero files, or no files matched the specified value for the optional parameter EXT=.

OUTPUT DATA SET STRUCTURE

Variable	Description
BASEFILE	Name of file (base file name only, without path)
PATHNAME	Full path name of file including directory name
FILE_SEQ	Sequence number of the file

The following variables are added to the data set only when ATTRIBS = Y is requested:

Variable	Description	Operating system
Owner_Name	File owner's ID	UNIX
Group_Name	Permission group	
Access_Permission	Permission string	
Last Modified	Date file created/modified	
File_Size__ Bytes	File size in bytes	
RECFM	Record format	Windows
LRECL	Logical record length	
File_size__bytes	File size in bytes	
Last_Modified	ddMONyyyy:HH:MM:SS	
Create_Time	ddMONyyyy:HH:MM:SS	

Table 2. Output data set structure

Sample Output

Obs	basefile	file_seq	Pathname	Owner_ Name
1	ae.xpt	1	/home/goulding/Experis/xptsdtm/ae.xpt	goulding
2	dm.xpt	2	/home/goulding/Experis/xptsdtm/dm.xpt	goulding
3	ds.xpt	3	/home/goulding/Experis/xptsdtm/ds.xpt	goulding
4	ex.xpt	4	/home/goulding/Experis/xptsdtm/ex.xpt	goulding
5	lb.xpt	5	/home/goulding/Experis/xptsdtm/lb.xpt	goulding

Obs	Group_ Name	Access_ Permission	Last_Modified	File_ Size_ bytes_
1	users	rw-rw-r-x	Wed Oct 9 11:28:35 2013	110400
2	users	rw-rw-r-x	Wed Oct 9 11:28:36 2013	6800
3	users	rw-rw-r-x	Wed Oct 9 11:28:36 2013	34800
4	users	rw-rw-r-x	Wed Oct 9 11:28:37 2013	11680
5	users	rw-rw-r-x	Wed Oct 9 11:28:39 2013	3775600

Output 1. Output data set created by the macro (running on HP-UX)

CODING EXAMPLE - USING THE MACRO'S RESULT TO DRIVE SUBSEQUENT PROCESS

In this scenario, we are required to process a directory full of SAS® version 5 transport files.

Name	Date modified	Type	Size
ae.xpt	10/9/2013 2:28 PM	SAS Xport Transport File	108 KB
dm.xpt	10/9/2013 2:28 PM	SAS Xport Transport File	7 KB
ds.xpt	10/9/2013 2:28 PM	SAS Xport Transport File	34 KB
ex.xpt	10/9/2013 2:28 PM	SAS Xport Transport File	12 KB
lb.xpt	10/9/2013 2:28 PM	SAS Xport Transport File	3,688 KB

Figure 1. Sample directory contents to be processed

The `dir_contents` macro is called to produce a data set with the names of all files having the target extension, `.xpt`.

```
/* first call the macro to produce the data set: work.xpt_sdtm */
%dir_contents(dir=/home/goulding/Experis/xptsdtm, ext=xpt, dsout=xpt_sdtm);
```

After the macro completes, the next section of the code – within a macro loop – runs a PROC COPY step once for each file, to convert each transport file into a standard SAS® data set. To drive the loop, the pathname values are loaded into a series of macro variables. (The global macro variable, `&_dir_fileN`, was created within `dir_contents`.)

```
/* populate a series of macro variables that contain the file names */
proc sql noprint;
select pathname into :path1 thru :path&_dir_fileN
  from xpt_sdtm;
quit;

/* process the names from the data set within a macro loop */
%macro run_loop;
  %do i = 1 %to &_amp;_dir_fileN;

    libname xlib xport "&path&i.";

    proc copy in=xlib out=work;
    run;

  %end;
%mend run_loop;
%run_loop
```

TECHNICAL DETAILS AND LIMITATIONS

The macro has one main data step that uses the functions listed below to first identify the number of files within the directory, and then builds the output SAS® data set from the directory contents.

- DOPEN to open the specified directory file
- DNUM to return the number of files that are in the directory
- DREAD to retrieve the name of each individual file
- DCLOSE to close the directory file after all file names are processed

Optionally, if the calling program specifies ATTRIBS = Y, then the macro performs a secondary data step that uses the functions listed below to access the available file attributes; these are then transposed and merged onto the primary data set as additional variables, one variable per file attribute.

- FOPEN to open each individual file
- FOPTNUM to return the number of information items (attributes) that are available about each file
- FOPTNAME to return the name for each attribute
- FINFO to return the attribute's value
- FCLOSE to close the file after attributes are obtained

The macro has a few limitations to be noted. If the requested directory itself contains subdirectories, these will be ignored – there is no recursive capability for stepping through multiple nested levels of subdirectories. The macro will accept only one extension passed on the EXT= parameter. If there is a need to filter on multiple extensions, you would need to call the macro once for each extension and set the results together.

CONCLUSION

This paper has presented a macro that can be applied in a variety of situations, as a tool that extends basic SAS® functionality for the benefit of clinical programmers. The macro encapsulates a basic set of functions without requiring the user to be familiar with the exact details of how these functions work.

Readers are encouraged to consider whatever uses might be the most advantageous in doing your own daily tasks.

REFERENCES

Hamilton, Jack. September 5, 2012. "Obtaining a List of Files in a Directory Using SAS Functions." *Western Users of SAS Software 2012 Proceedings*. Available at <http://www.wuss.org/proceedings12/55.pdf>.

Dilorio, Frank. September 26, 2010. "Building the Better Macro: Best Practices for the Design of Reliable, Effective Tools." *Southeast SAS Users Group Conference Proceedings*. Available at <http://analytics.ncsu.edu/sesug/2010/FF02.Diloriopdf.pdf>

ACKNOWLEDGMENTS

The author would like to thank the superb administrative professionals at Experis for their invaluable behind-the-scenes support. I would also like to thank the leadership team of our Experis Clinical Practice for their ongoing encouragement.

CONTACT INFORMATION

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

Name: Mike Goulding
Enterprise: Experis
Address: 5220 Lovers Lane
City, State ZIP: Portage, MI 49002
Work Phone: 734-277-0541
E-mail: michael.goulding@experis.com

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 trademarks of their respective companies.

APPENDIX – DIR_CONTENTS MACRO CODE

```

%macro dir_contents(
  dir=                /* directory name to process          */
, ext=                /* optional extension to filter on      */
, dsout=work.dir_contents /* dataset name to hold the file names */
, attribs=N          /* get file attributes? (Y/N )         */
);

%global _dir_fileN;
%local _syspathdlim _exitmsg _attrib_vars;

/* verify the required parameter has been provided. ;
%if %length(&dir) = 0 %then %do;
  %let _exitmsg = %str(E)RROR: No directory name specified - macro will exit.;
  %goto finish;
%end;

/* verify existence of the requested directory name. ;
%if %sysfunc(fileexist(&dir)) = 0 %then %do;
  %let _exitmsg = %str(E)RROR: Specified input location, &dir., does not exist - macro
will exit.;
  %goto finish;
%end;

/* set the separator character needed for the full file path: ;
/* (backslash for Windows, forward slash for UNIX systems) ;
%if &sysscp = WIN %then %do;
  %let _syspathdlim = \;
%end;
%else %do;
  %let _syspathdlim = /;
%end;

/*--- begin data step to capture names of all file names found in the specified
directory. ---*/

data &dsout(keep=file_seq basefile pathname);

  length basefile $ 40 pathname $ 1000 _msg $ 1000;

  /* Allocate directory */

  rc=FILENAME('xdir', "&dir");

  if rc ne 0 then do;
    _msg = "E" || 'RROR: Unable to assign fileref to specified directory. ' ||
sysmsg();
    go to finish_datastep;
  end;

  /* Open directory */
  dirid=DOPEN('xdir');
  if dirid eq 0 then do;
    _msg = "E" || 'RROR: Unable to open specified directory. ' || sysmsg();
    go to finish_datastep;
  end;

  /* Get number of information items */
  nfiles=DNUM(dirid);

```

```

do j = 1 to nfiles;
  basefile = dread(dirid, j);
  pathname=strip("&dir") || "&_syspathdlim." || strip(basefile);

  %if %length(&ext) %then %do;
    /* scan the final "word" of the full file name, delimited by dot character. */

    ext = scan(basefile,-1, '.');
    if ext="&ext." then do;
      file_seq + 1;
      output;
    end;
  %end;
  %else %do;
    file_seq + 1;
    output;
  %end;
end;

/* Close the directory */
rc=DCLOSE(dirid);

/* Deallocate the directory */
rc=FILENAME('xdir');

call symputx('_dir_fileN', file_seq);
finish_datastep:
if _msg ne ' ' then do;
  call symput('_exitmsg', _msg);
end;
run;

%if %upcase(&attribs)=Y and &_dir_fileN > 0 %then %do;

data _file_attr(keep=file_seq basefile infoname infoval);
  length infoname infoval $ 500;
  set &dsout.;

  /* open each file to get the additional attributes available. */
  rc=filename("afile", pathname);
  fid=fopen("afile");

  /* return the number of system-dependent information items available for the
external file. */
  infonum=foptnum(fid);

  /* loop to get the name and value of each information item. */
  do i=1 to infonum;
    infoname=foptname(fid,i);
    infoval=finfo(fid,infoname);
    if upcase(infoname) ne 'FILENAME' then output;
  end;

  close=fclose(fid);
run;

/* transpose each information item into its own variable */
proc transpose data=_file_attr out=trans_attr(drop=_) ;
  by file_seq basefile ;
  var infoval;
  id infoname;
run;

```

A Macro to Produce a SAS® Data Set Containing the List of File Names Found in the Requested Windows or UNIX Directory, continued

```
proc sql noprint;
  select distinct name into : _attrib_vars separated by ', '
    from dictionary.columns
    where memname='TRANS_ATTR' and upcase(name) not in('BASEFILE', 'FILE_SEQ')
  order by varnum;
quit;

/* merge back the additional attributes to the related file name. */
data &dsout.;
  merge &dsout. trans_attr;
  by file_seq basefile;
run;

proc datasets nolist memtype=data lib=work;
  delete _file_attr trans_attr;
run;
quit;
%end;

%if %length(&_amp;exitmsg) = 0 %then
%let _amp;exitmsg = NOTE: &dsout created with &_dir_fileN. file names ;

%if %length(&_amp;ext) %then
%let _amp;exitmsg = &_amp;exitmsg where extension is equal to &_amp;ext.;

%let _amp;exitmsg = &_amp;exitmsg from &dir..;

%finish:
%put &_amp;exitmsg;

%if %length(&_amp;attrib_vars) ne 0 %then %do;
  %put;
  %put NOTE: File attributes were requested and have been added to &dsout.. Variable
names are &_amp;attrib_vars.;
%end;

%mend dir_contents;
```