Paper 74-28

So Many Files, So Little Time (or Inclination) to Type Their Names: Spreadsheets by the Hundreds

Francis J. Kelley, University of Georgia, Athens, GA
jkelley@uga.edu

Abstract. It may often be the case that many files, all with the same structure, may need to be read for subsequent analysis. When this is some (small) number, it is trivial to enter the specific names. However, should an entire directory, possibly hundreds of files, have to be used, then anything that avoids having to type in each specific file name must be considered. Fortunately, SAS Software® provides some very useful tools for doing this via the DATA step. Now suppose the data cannot be read with the DATA step, but must be read using PROC IMPORT (an Excel™ spreadsheet, for example). Is it now necessary to enter the names of each file to be read? No, a method similar to that used for text data may be employed. There are simple, useful tools that allow this, though many new SAS users are unaware of them (and some of the older ones may have overlooked them).
NOTE: this requires SAS/Access for PC File Formats®.

Text example 1. Suppose we have three files:

```
FileA.dat
-----------------------
A6037    102.9      110.1
A8810    111.2      100.2
C2319    113.6      119.1

FileB.dat
-----------------------
A7021    100.8      115.4
C2309    121.0      113.6
B5803    108.4      112.4

FileC.dat
-----------------------
B2974    117.2      115.7
B7012    103.8      114.6
B0264    110.6      107.3
```

It is trivial to specify each of these on the **FILENAME/INFILE** statements, but there is another way that is more useful; the **FILEVAR=** option on the INFILE statement. Here is one way we might use that:

```
options pageno=1 ;
data test;
    length inf_var cur_f_var
      $ 100 ;
    infile CARDS ;
* Read the FileNames from CARDS;
    input @1 inf_var $ ;
    infile DUMMY  end=done
        filevar=inf_var
        filename=cur_f_var;
* Read the Data from the FILEVAR ;
* Set the END= and use 'DO WHILE';
        do while (not done) ;
            input ID $ Score1 Score2 ;

                output ;
            end ;
cards;
"k:\TestData\filea.dat"
"k:\TestData\fileb.dat"
"k:\TestData\filec.dat"
;
proc print   data=test ;
title "Reading Several Files";
run;
```

This is a very nice example and the outline for it is found in the *SAS OnLine Documentation* for Version 8.

The original problem still remains, of course: some large number of files all to be read with PROC IMPORT.

Text example 2. Using the same data files, but not reading their names from "Cards". The file names could be read from an external file, but that would not help in what we want.

SAS provides the (unnamed) PIPE access method for all systems except OS/390 (it is available there via "UNIX System Services"). Using this, it is possible to obtain the directory listing. In the Microsoft Windows™ environment, this is done with the venerable "**dir**" command.
Actually, to avoid unneeded parsing, a wildcard ('*') and the '/b' ('bare') switch are used to reduce the list to just what is needed. Here is the new version:

```
options pageno=1 ;

%let my_dir = k:\TestData\;

filename dir_list pipe
    "dir &my_dir.*.dat /b" ;
data test;
    length  inf_var cur_f_var
        fil2rd $ 100 ;
* Read the directory from PIPE ;
    infile dir_list ;
    input @1 inf_var $ ;
        fil2rd=
          "&my_dir"||trim(left(inf_var));
* read the data from the FILEVAR;
* set END= and use 'DO WHILE';
    infile DUMMY  end=done
        filevar=fil2rd
        filename=cur_f_var;
        do while (not done) ;
            input ID $ Score1 Score2 ;
             output ;
        end ;
```

```
proc print   data=test ;
title  "Reading Several Files";
title2 "Directory: &my_dir";
run;
```

This is much the same as the previous example, but note that the Macro variable "**my_dir**" has been defined. This is the directory where the various "dat" files reside. Setting up a macro variable will make several things easier:

1. The pipe may be defined using the macro variable (and always remember to use "" when quoting macro variables):
   ```
   filename dir_list pipe
   "dir &my_dir.*.dat /b" ;
   ```
   At execution this will become
   ```
   filename dir_list pipe
   "dir k:\TestData\*.dat /b" ;
   ```
   and the list of all .DAT files will be made.

2. Because only the filenames are produced (e.g "filea.dat"), the path (defined in the "**&my_dir**" macro variable) is used to produce the full path/filename for the FILEVAR= variable (FIL2RD) (and when dealing with character strings, particularly with concatenation, it is a good idea to be sure there are no leading – or trailing – blanks – hence the "trim").

The variables named with both FILEVAR= and FILENAME= are not written to the output SAS data set, but the variable read from the PIPE was not FIL2RD, so it will be preserved. This can be very useful as it allows the data to be tagged with the name of the file from which it originated.

Here is what the output looks like:

Reading Several Files
Directory: k:\TestData\

inf_var	ID	Score1	Score2
FILEA.DAT	A6037	102.9	110.1
FILEA.DAT	A8810	111.2	100.2
FILEA.DAT	C2319	113.6	119.1
FILEB.DAT	A7021	100.8	115.4
FILEB.DAT	C2309	121.0	113.6
FILEB.DAT	B5803	108.4	112.4
FILEC.DAT	B2974	117.2	115.7
FILEC.DAT	B7012	103.8	114.6
FILEC.DAT	B0264	110.6	107.3

PROC IMPORT example 1. With this, the basics of how to read the Excel Spreadsheet have been described, but now they must be implemented. A significant difference is the lack of anything like the DATA step's FILEVAR=. This allowed the cycling through all the data files. There is nothing like that in Proc IMPORT. Fortunately, it may be done with a very simple Macro.

First, however, it is probably best to review how a single Spreadsheet is read:

```
options pageno=1 ;

proc import
      datafile='k:\TestData\FileA.xls'
      out=excel_test
      dbms=excel2000
      replace;
    getnames=yes ;

proc print   data=excel_test noobs;
title  "Reading One File";
run;
```

The output will look like this:

Reading One File

ID	Score1	Score2
A6037	102.9	110.1
A8810	111.2	100.2
C2319	113.6	119.1

PROC IMPORT example 2. Looping through multiple calls to the procedure will require the use of the Macro facility. Still, the basics will be much as we have already seen. The full code for this is included at the end of this paper.

As before, a PIPE access method will be used to read the contents of the specified directory – again named in the Macro Variable &MY_DIR. Note that this path (&MY_DIR) will always end with a "\" as it will be concatenated with the actual filenames to produce the complete path/filename specification.

Unlike the Text example, the filenames, path/filename combinations, and lastly the total number of files ("&TOT_FILZ") are all saved as Macro Variables in a "DATA _NULL_" step. As a final part of the setup, a dummy record is written to the output dataset; it consists of one observation with no variables.

The Macro Variables.
In the DATA _NULL_ step, two character variables, F_NAME1 and F_NAME2 are defined. F_NAME1 consists of the names of the files in the directory (in the example, all the .XLS files). F_NAME2 is the full path/filename for each F_NAME1. A counter variable, K, is maintained and a "character" version (CK) is also set up. Individual macro variables for each of F_NAME*n* are set up dynamically with the CALL SYMPUT function. The values of F_NAME1 are saved into &F1, &F2 and so on. Likewise, the values of F_NAME2 are saved into &G1, &G2 and etc. At the end, the final value of CK is saved as &TOT_FILZ.

And now the Macro loop begins:

%DO i = **1** %TO &TOT_FILZ;
Using the Macro variable &I as an index, it is possible to retrieve the specific path/filenames for each spreadsheet and save then to "__TEST":

```
proc import  datafile="&&G&i"
```

```
            out=__test
            dbms=excel2000
            replace;
    getnames=yes ;
```

A word about "&&G&I":
The SAS Macro processor will first resolve this to "&G1" and only then will it resolve to "*_f_name2_*" – whatever that might be. Multiple ampersands (&) can seem more intimidating than they really are.

The file written by IMPORT (__TEST) is now concatenated to the output dataset (&DSOUT). The dummy record was written in a previous step just to avoid a contortion at this point when I=1 and &DSOUT would have had to be created and not just appended to. Here too, the name of the file being read (saved as macro variable &F1 …) is added to the dataset. This is optional, but may be quite handy in working with the complete data file later.

This will continue as long as there are files to read, and then the Macro loop is exited.

To get rid of the dummy record written to &DSOUT a single step is called for:

```
data &dsout ;
   set &dsout;
   if _N_ = 1 then delete;
run;
```

Now the macro ends.

Conclusion. Programmers should not do what their tools can do (better) for them. This example was based on a real case in which research data had to be combined from several hundred spreadsheets. Attempting to convert these individually would have been tedious and error-prone. Writing a program to do it was fun and easy. "Fun and easy" wins over "tedious and error-prone", and provides a way of expanding one's knowledge. In developing this paper, an effort was made to demonstrate how one moves from reading text data in a DATA step to using a Macro to implement a similar methodology for reading the spreadsheet data in a PROC.

While macros can be overused yet (perversely) poorly-understood or documented, and when misused may lead to serious problems, they can also be of great benefit to the SAS programmer. Even the simple use of macro variables ("Text Example 2") can make programming easier and the host of handy options on the INFILE statement (in this case, FILEVAR=) are of great use in developing programs that have great power, yet are relatively simple.

As a final note, although the emphasis has been on files with a similar/identical organization, the macro shown could be adapted to spreadsheets with very different organizations. That is beyond the scope of this paper, so the "proof is left to the interested student".

Trademarks.

References.

-*SAS OnlineDoc® Version Eight*. Copyright (c) 1999 SAS Institute Inc., Cary, NC, USA.
-*SAS System Help for Version 8.2*. Copyright SAS Institute Inc., Cary, NC
-SAS-L (sas-l@listserv.uga.edu) on-line SAS discussion group.

Author Contact Information.

Questions and comments are encouraged!

Francis Joseph Kelley
Senior Consultant, Enterprise IT Services
University of Georgia
Athens, GA 30602-1911
jkelley@uga.edu

Program Code:

```
options pageno=1 ;
options mprint ;
%let my_dir = k:\TestData\ ;

filename xcl_fil pipe
    "dir &my_dir.*.xls /b";

%MACRO  Mult_Fil(PIPEin=,DSout=);
    %LOCAL i ;
            data _NULL_ ;
                infile &PIPEin end=last;
                retain k 0 ;
                k + 1 ;
                length f_name1 f_name2 $ 60 ;
                input f_name1 $ ;
```

```
                f_name2 = "&my_dir"||trim(left(f_name1));
                ck = compress(put(k,3.));
                    * This will produce both the filename  ;
                    * and the path/filename                ;
                call symput('F'||ck,trim(left(f_name1)));
                call symput('G'||ck,f_name2);
                    * Now, a count of the number of files  ;
                if last then call
                    symput('TOT_FILZ',ck);
            run;
            * Create Dataset with Dummy Record ;
            data &dsout ;
            run;
    %DO i = 1 %TO &TOT_FILZ;

            proc import  datafile="&&G&i"
                    out=__test
                    dbms=excel2000
                    replace;
                getnames=yes ;

            data __test ;
                set __test;
                    * Save the name of the file this    ;
                    * data came from                    ;
                file = "&&F&i";

            data &dsout ;
                    * Stack the files                   ;
                set &dsout __test ;
    %END ;

            data &dsout ;
                set &dsout;
            * remove Dummy Record ;
                if _N_ = 1 then delete;
            run;

%MEND Mult_Fil ;

%mult_fil(PIPEin=xcl_fil,DSout=perm)
proc print   data=&syslast;
title  "Reading All Files";
run;
```

Example Output:

```
                        Reading All Files                          1
                                    14:21 Thursday, October 1, 2002

           Obs    ID       Score1    Score2      file

            1     A6037     102.9     110.1    FileA.xls
            2     A8810     111.2     100.2    FileA.xls
            3     C2319     113.6     119.1    FileA.xls
            4     A7021     100.8     115.4    FileB.xls
            5     C2309     121.0     113.6    FileB.xls
            6     B5803     108.4     112.4    FileB.xls
            7     B2974     117.2     115.7    FileC.xls
            8     B7012     103.8     114.6    FileC.xls
            9     B0264     110.6     107.3    FileC.xls
```