

The Demystification of a Great Deal of Files

Chao-Ying Hsieh, Southern Company Services, Inc. Atlanta, GA

ABSTRACT

Our input data are sometimes stored in external flat files rather than in a traditional database environment. This creates tedious work if programmers need to read a large amount of input files from multiple locations. This paper will address a solution to this issue that uses the SAS® macro and DREAD function. Additionally, the paper will also address further applications for using file name information to validate data.

INTRODUCTION

A project in our group requires us to extract contract information from another department. The contract information is stored as an individual flat file (ASCII text file) for each individual account, and these flat files are located in different folders depending on when the contracts are established or adjusted. The number of flat files can possibly exceed over two thousand.

The names of these files start with customer's account number and end with specific file extension (first two characters with 'cb' and the third character with an alphanumeric field, such as cb1, cb0 or cbl). For example, if an account number is 1100, its contract information is stored as a file called 1100.cb1. Usually, the contracts are setup in the beginning of the year, but customers can modify the contracts anytime throughout the year. When a contract is modified, instead of modifying the original file, an additional file is added to the folder of the month. For example, if an account changes its contract in both March and June, there are three files for this account. One in the 'Originals' folder, which is created in the beginning of the year, another one in the 'March' folder, and the last one in the 'June' folder. Sometimes, there will be multiple folders representing the same month. For example, there are two folders for January ('Originals' and 'January'). Folders 'May', 'May Corrected', and 'May Corrected2' all store the files for May (figure 1). There are also hierarchy ranking relationships between these folders, but the names and hierarchy ranking can also change every year. At that time, our clients have other files besides the contracts stored in the same folders.

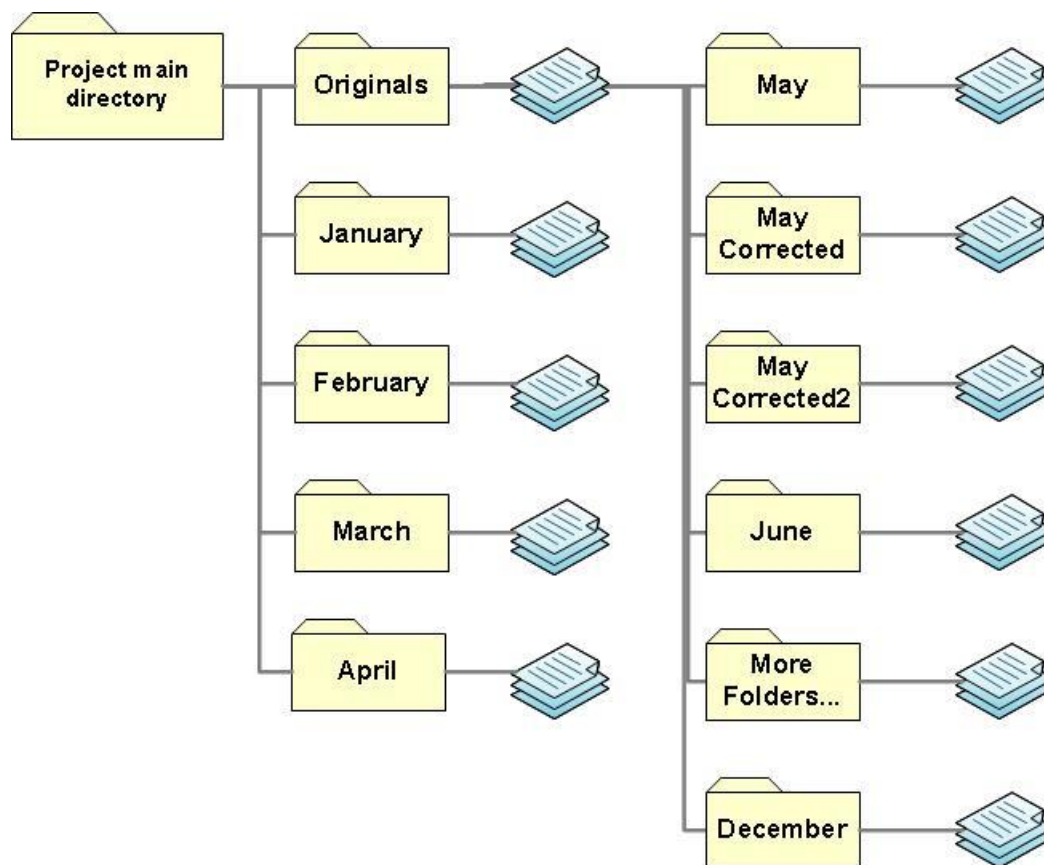


Figure 1. Diagram of Client's Project Directory

In order to get the contract information correctly, I need to find out which accounts have which contracts, where those contract files are stored, whether there are any adjustments in their contracts, and when these adjustments happened.

SAS FUNCTIONS FOR EXTERNAL FILES

There are many SAS functions for manipulating external files. They can be used within the data step or combined with macro function %sysfunc in a macro process.

SAS Function Name	Syntax	Purpose
FILENAME	FILENAME(fileref <,file-name> <,device-type> <,'host-options'> <,dir-ref>)	Assigns or deassigns a fileref to an external file, directory, or output device.
DOPEN	DOPEN(fileref)	Opens a directory, and returns a directory identifier value
DCLOSE	DCLOSE(directory-id)	Closes a directory that was opened by the DOPEN function
DNUM	DNUM(directory-id)	Returns the number of members in a directory
DREAD	DREAD(directory-id,nval)	Returns the name of a directory member

Table 1. SAS functions for manipulating external files

SAS(R) 9.4 Functions and CALL Routines: Reference, Fourth Edition

<https://support.sas.com/documentation/cdl/en/lfunctionsref/67960/HTML/default/viewer.htm#n0kiirjm2dlaz9n1d9uci86quxf3.htm>

There is a great example code from SAS online document about getting the last member name from a directory, and place it in macro variable LSTNAME:

```
(1) %let filrf=mydir;
(2) %let rc=%sysfunc(filename(filrf,physical-name));
(3) %let did=%sysfunc(dopen(&filrf));
(4) %let lstname=;
(5) %let memcount=%sysfunc(dnum(&did));
(6) %if &memcount > 0 %then
    %let lstname=%sysfunc(dread(&did,&memcount));
(7) %let rc=%sysfunc(dclose(&did));
```

Source:

<http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000209687.htm>

Line by line description of the code:

- (1) Assign MYDIR to a macro variable FILRF.
- (2) Use FILENAME function to associate MYDIR to a physical path name.
- (3) Use DOPEN function to open MYDIR directory and it returns a directory ID, DID.
- (4) Initiate a macro variable LSTNAME.
- (5) Use DNUM function to get the total number of member to macro variable MEMCOUNT.
- (6) Use DREAD function to get the last member name and assign it to the macro variable LSTNAME.
- (7) Close directory with the function DCLOSE.

MULTIPLE DIRECTORIES

The example code from SAS only reads the information from one directory. My data locate in multiple folders. To read the files from multiple directories, Jack Hamilton had a paper in 2012 Proceedings of Western Users of SAS Software. He talks about using MOPEN function to find directories recursively. I cannot use his technique here because the names of the directories I am working on have specific meanings. The names of the directories and hierarchy ranking of the folders can also change every year. I need to find a way to let my program recognize multiple folders' names, the hierarchy ranking and the month without changing my code every year. I decide to create a SAS data set to include the names of the directories, month and hierarchy ranking. Then I can use that as an input information data set.

Table 2 lists a sample of the input data set. Variable NAME contains subfolder name. Variable RANK has the hierarchy ranking number. Variable REAL_MONTH is the month that the subfolder represents.

name	Rank	real_month
Originals	0	1
January	1	1
February	2	2
March	3	3
April	4	4
May	5	5
May Corrected	5.1	5
May Corrected 2	5.2	5
June	6	6
July	7	7
August	8	8
September	9	9
October	10	10
November	11	11
December	12	12

Table 2. Input Data for Subdirectory Names

In this case, there are multiple folders for January and May. The hierarchy number from variable RANK can help me select the contract data if the same account has multiple files in different folders. I can now use this input data set to create a series of macro variables to capture the names of the subdirectories, the ranking numbers, and the months that they represent. At the end of data file, macro variable TOTN is created with the total number of records.

```

data _null_;
set a end=eof;
retain n;
n+1;
call symput('mon' || strip(put(n,2.)), put(rank,2.));
call symput('real' || strip(put(n,2.)), put(real_month,2.));
call symput('f' || strip(put(n,2.)), strip(name));

if eof then call symput('totn', strip(put(n,2.)));
run;

```

The corresponding values of macro variables are listed in table 3.

name	rank	real_month
&f1= Originals	&Mon1= 0	&real1= 1
&f2= January	&Mon2= 1	&real2= 1
&f3= February	&Mon3= 2	&real3= 2
&f4= March	&Mon4= 3	&real4= 3
&f5= April	&Mon5= 4	&real5= 4
&f6= May	&Mon6= 5	&real6= 5
&f7= May Corrected	&Mon7= 5.1	&real7= 5
&f8= May Corrected 2	&Mon8= 5.2	&real8= 5

name	rank	real_month
&f9= June	&Mon9= 6	&real9= 6
&f10= July	&Mon10= 7	&real10= 7
&f11= August	&Mon11= 8	&real11= 8
&f12= September	&Mon12= 9	&real12= 9
&f13= October	&Mon13= 10	&real13= 10
&f14= November	&Mon14= 11	&real14= 11
&f15= December	&Mon15= 12	&real15= 12

Table 3. List of Corresponded MACRO Variables

After putting subdirectory information into macro variables, I build a macro process to read in each subdirectory to get the file name and output and append the file to a data set called ALLFILES.

```
%macro rr;

proc datasets lib=work;
delete add allfiles;
quit;
%do m= 1 %to &totn;

    %let filrf=mydir;
    %let rc=%sysfunc(filename(filrf,"&datapath.&f&m\"));
    %let did=%sysfunc(dopen(&filrf));
    %let totf=%sysfunc(dnum(&did));

    %if %eval(&totf) > 0 %then
    %do i=1 %to &totf;

        %let fname=%sysfunc(dread(&did,&i));

        data add;
        length file_name $200 folder $20 ext $3;
        folder="&f&m";
        file_name="&fname";
        account_number=input(scan(file_name,1), 10.);
        ext=reverse(scan(reverse(strip(file_name)),1,'.'));
        rank=&mon&m;
        real_month=&real&m;
        run;

        proc append base=allfiles data=add;
        run;

    %end;
    %let rc=%sysfunc(dclose(&did));
%end;

%mend rr;

%rr
```

Final data ALLFILE includes account number (variable KY_BA), month (variable REAL_MONTH), hierarchy ranking (variable RANK), subfolder name (variable FOLDER), file name (variable FILE_NAME) and file extension (variable EXT).

File_name	Folder	ext	Rank	Account_number	real_month
0000000001.cb0	Originals	cb0	-1	1	1
0000000002.cb0	Originals	cb0	-1	2	1
0000000003.cb0	Originals	cb0	-1	3	1
0000000004.cb0	Originals	cb0	-1	4	1
0000000005.cb0	Originals	cb0	-1	5	1

Table 4. Example Records of Data ALLFILES

APPPLICATION AND MORE

There are advantages to store file information in a data set. Not only can I use the information to adjust customers' contract data later, but I can also utilize them to validate the process. According to my client, contract files have the extension started with 'cb'. Since my clients store files other than contracts in the same location, those non-contract files need to be excluded. These non-contract type files should then be extracted to data set CHK. The contract files information are kept in data set ALLFILES. The file extensions of both contract and non-contract files are printed out as two listings. These two listings will serve as the validation reports. The analyst can check on the listings and verify whether a file is categorized properly. It is a way to guarantee that the non-contract files are excluded correctly.

```
data chk allfiles;
set allfiles;
if ky_ba = . or upcase(substr(ext,1,2)) ne 'CB' then output chk;
else output allfiles;
run;
```

```
proc sql;
select distinct folder, ext
from chk
order by rank;
select distinct ext
from allfiles;
quit;
```

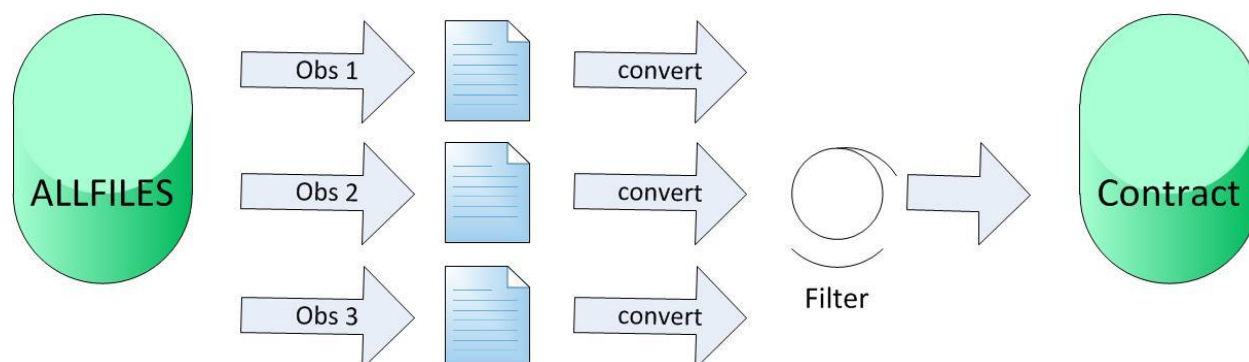
folder	ext
Originals	LOG
Originals	Upd
Originals	Ink
Originals	pmd
January	BD0
January	BDH
January	bd0
February	bd0
February	Xml

Table 5. Validation Report for File Extension of Excluded Files

ext
CB0
CBL
cb0
cb1

Table 6. Validation Report for File Extension of Contract Files

Once I get the filename information, I can use it to build another macro process to read in every flat file that contains the contract information. Month and hierarchy ranking can also be used to adjust the contract data if the customers have multiple contract files. In the end, the correct contract data will be created for each customer.

**Figure 2. Diagram of Macro Process to Create Final Contract Data**

CONCLUSION

When we deal with a large amount of non-traditional data, efficiency and accuracy become critical. Combining SAS function and macro is a powerful technique to achieve both. The run time of searching for files from my program is less than a minute. There are also validation reports available to prevent missing new contract file types or typos in the file names too.

REFERENCES

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

SAS® 9.4 Functions and CALL Routines: Reference, Fourth Edition Available at <https://support.sas.com/documentation/cdl/en/lefunctionsref/67960/HTML/default/viewer.htm#n0kiirjm2dlaz9n1d9uci86quxf3.htm>.

SAS® 9.2 Language Reference: Dictionary, Fourth Edition Available at <http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000209687.htm>.

ACKNOWLEDGMENTS

The following people contributed extensively to the development of this paper: Bob Bolen and Celia Wang at Southern Company and Sarah Tsai. Their support is greatly appreciated.

CONTACT INFORMATION

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

Chao-Ying Hsieh
 Southern Company Services, Inc.
 214 Ralph McGill BLVD
 Atlanta, GA 30308
 E-mail: chsieh@southernco.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.