Paper 035-30

Automatically Process a Varying Number Of New Data Files From a "Data Directory"

Michael A. Raithel, Westat, Rockville MD

Abstract

Imagine a situation in which you are sent dozens of new data files, by a varying number of users, which must be processed on a daily basis. Perhaps the users are at other locations and are FTP-ing the data files to a designated "data directory." Perhaps the users are within your organization and are writing the data files from other operating systems to a data directory allocated via a Samba share drive. Perhaps the users are all on a shared server within your organization and are writing new data files to a data directory on the same system. Whatever the case, this scenario usually generates a lot of maintenance to the SAS program that processes the data files. When new users are added, you must modify your program to allocate and process their specific files. When users are removed, you must modify your program so that it does not attempt to allocate and process their data files. And, when a given file is not present for one reason or another, you must add logic so that your program does not abend.

This paper presents a methodology of automatically processing a varying number of new data files from a data directory. The paper presents three straightforward SAS programs that make the process simple and error-free. After reading this paper, you should be able to tailor the three SAS programs to process data files in your own organization.

Introduction

One of the most frustrating aspects of creating a system to process a varying number of data files is the constant maintenance of the SAS programs. Since you are never certain exactly how many files will show up in the data directory, or the exact names of the files, you must be constantly changing your SAS program. This may be ameliorated to some extent when you have great communication with the people who are creating the data files. They may be diligent about letting you know that a new file by a new user with a given name will be transmitted to the data directory beginning on a given date. However, you are still left with modifying your SAS program to read that specific file. Lapses in communication will mean that you have either missed a new data file, or that your SAS program abended because a file that it was expecting to process was never sent to the data directory.

These problems can be mitigated by taking a more global view of the data directory. The SAS programs in this paper treat the data directory as a "location". They simply process all of the data files that are stored in that location when the programs are invoked. The SAS programs do not particularly care about the number of data files in the directory or the data file names. Armed with the directory name, they simply access the directory, process each data file, and then delete each data file when finished processing it. This global view facilitates simplified SAS program maintenance and limits mistakes made due to file naming conventions or files not arriving in the data directory as expected.

The three SAS programs that make the processing of a varying number of data files from a data directory possible can be found in **Appendix A**. The following sections of this paper describe each SAS program in detail. The programs were set up to run under SAS in the Microsoft® Windows environment, so you should keep that in mind when looking at them. You may easily modify the systems interfaces of these programs so that they may run in a Unix, Linux, VAX, OS/2, or z/OS environment.

Driver_Program.sas

This SAS program is a "driver" program that is used to execute the other programs. When you decide to add or remove data directories, you simply add or delete them from the bottom of this program. All of the editing should be to the driver once you have the three programs in production. This strategy keeps the maintenance to this application simple and predictable.

The job of Driver_Program.sas is to %INCLUDE the other SAS programs necessary to get the work done, and then to execute them for the specified data directories. The driver program should be scheduled to run at a time—or times—when the greatest number of data files are likely to have been placed in the data directories. On the Windows operating system, you can use the Windows Job Scheduler to schedule the **.bat** program to run on a cyclic basis.

A sample **.bat** file to run Driver_Program.sas on a Windows server might look like this:

```
"H:\Program Files\SAS Institute\SAS\V8\SAS.EXE"  -icon -noterminal -nosplash
-noxwait -noxsync
-CONFIG "C:\Program Files\SAS Institute\SAS\V8\Sasv8.cfg"
-SYSIN "Q:\programs\Driver_Program.sas"
-LOG  "Q:\programs\Driver_Program.log"
```

The **.bat** file, above, will execute the Driver_Program.sas program located in the Q:\programs directory and store the SAS log file in the same directory. Though it is not illustrated, above, this **.bat** file is set up to run once a day (in this instance) by the Windows scheduler for this particular application.

In the driver program, the first %INCLUDE statement includes the *Process_Flat_Files.sas* SAS program. This is the program that will process individual data files. Since the *Process_Flat_Files.sas* program only contains the PROCESS SAS Macro, it will be included at this point, but not executed.

The second %INCLUDE statement includes the *Identify_Flat_Files.sas* SAS program. That program contains the *Identify_Flat_Files.sas* SAS Macro and will not execute it at the point in time when it is %INCLUDE-ed. The *Identify_Flat_Files.sas* program reads all of the data file names in a specified directory and creates calls to the IDENTIFY SAS Macro located in the *Process_Flat_Files.sas* program.

At the bottom of the driver program, you will find the calls to the IDENTIFY SAS Macro. Each invocation of the IDENTIFY SAS Macro specifies: 1.) the directory where the permanent SAS data set used to store data extracted from the data files resides, 2.) the name of the data directory that is to be processed. By invoking the IDENTIFY SAS Macro, the files of a specified directory are processed. They are identified by the IDENTIFY SAS Macro, and then processed by the READFLAT SAS Macro, since both Macros were previously %INCLUDE-ed.

Identify_Flat_Files.sas

The *Identify_Flat_Files.sas* program contains the IDENTIFY SAS Macro. This program reads all of the data file names in a directory and creates calls to the READFLAT SAS Macro, located in the *Process_Flat_Files.sas* program. The IDENTIFY SAS Macro accepts two parameters:

- SASLIB – This is the full path name of the production SAS data library. This value will be coded into the Macro call for the READFLAT SAS macro that is the output of this program.

- DIRNAME – This is the full path name of the data directory that contains the data files.

An invocation of the IDENTIFY SAS Macro might look like this:

%IDENTIFY(Q:\data,\\system1\input\datadir);

In the Macro invocation, above, "**Q:\data**" is the production SAS data library where information processed from the data files will be stored**. \\system1\input\datadir** is the data directory where the individual data files can be found.

The IDENTIFY SAS Macro begins execution by piping in the output from executing a Windows system command via a FILENAME statement:

```
filename dircmd pipe "dir /b &DIRNAME\*.txt";
```

That command, "dir /b", is executed to provide a list of all data files in the data directory specified by the &DIRNAME Macro variable that is passed to the IDENTIFY SAS Macro. The FILENAME statement executing the "dir /b" command only accepts files with a suffix of ".txt". A typical output from execution of the FILENAME statement might look like this:

store22_01212005.txt
store55_01212005.txt
store64_02212005.txt
store97_02212005.txt

In the example, above, there were four .txt data files in the specified data directory. All four file names were passed, in turn, to the SAS program via the execution of the FILENAME statement.

A temporary flat file, with a fileref of "HOLDMACS", is used to store the output of the IDENTIFY SAS Macro. The flatfile will contain one entry for each data file name that was input via the "dir /b" command piped in via the FILENAME statement. Each entry written out to HOLDMACS will be a syntactically correct SAS call of the READFLAT SAS Macro. So, for the four files listed in the example, above, the following would be written to the HOLDMACS temporary flat file:

%READFLAT(Q:\data,Q:\system1\datadir\store22_01212005.txt);
%READFLAT(Q:\data,Q:\system1\datadir\store55_01212005.txt);

%READFLAT(Q:\data,Q:\system1\datadir\store64_01212005.txt);
%READFLAT(Q:\data,Q:\system1\datadir\store97_01212005.txt);

The temporary flat file is created by the single DATA _NULL_ step in the IDENTIFY SAS Macro. The DATA step reads each data file name piped in through the DIRCMD fileref, one at a time. It then constructs a call of the READFLAT SAS Macro using:

- &SASLIB – The permanent SAS data library Macro parameter passed in to the IDENTIFY SAS Macro
- &DIRNAME – The full path to the data directory passed in to the IDENTIFY SAS Macro
- The name of the specific data file read in via the DIRCMD fileref, which piped in the "dir /b" Windows system command.

After the READFLAT SAS Macro call is fully constructed in variable OUTLINE for a given data file, the DATA step writes it out to the HOLDMACS flat file. Then, the DATA step processes the next data file name in the same way.

The final act of the IDENTIFY SAS Macro is to %INCLUDE the HOLDMACS file. By doing so, it executes, one-by-one, each distinct call to the READFLAT SAS Macro that it previously created.

Process_Flat_Files.sas

The *Process_Flat_Files.sas* program contains the READFLAT SAS Macro, which is the workhorse of the entire data file processing procedure. The READFLAT SAS Macro accepts two parameters:

- PRODFILE - This is the full path filename of the SAS data library where the production SAS data set used to store the processed data files resides.

- DATAFIL - This is the full path filename of the individual data file that will be processed by this program.

The READFLAT SAS Macro begins with a FILENAME statement that allocates the data file that will be processed by this program. It uses the DATAFIL fileref and the &DATAFIL Macro variable which passes the full path of the data file to the program.

Once the data file has been allocated to the SAS program, a DATA _NULL_ step determines whether or not the file can be opened. This is necessary in organizations where it is possible that the data file could be in the process of either being created or being updated when the *Process_Flat_Files.sas* program executes. If the data file is in use, then a message is written to the SAS Log and the rest of the READFLAT SAS Macro program is skipped for this particular file. Otherwise, the program moves on to the next DATA step.

The next DATA step reads each line of the data file, extracts the pertinent information, and stores it into a Work SAS data set named NEWDATA. The example, in **Appendix A**, was left blank, because it will be different for each application. Whatever your data selection and processing logic may be, should be coded into the second DATA step in the READFLAT SAS Macro.

Following the DATA step that processes the data file is a PROC SQL step. The PROC SQL step determines if there are any observations in NEWDATA. This is necessary, because it is possible in certain circumstances for a data file to be empty, or for a data file to not contain valid data that you would like to have stored in the permanent SAS data set. The PROC SQL step stores the number of observations contained in NEWDATA in a new SAS Macro variable named NUMBROBS.

If NUMBROBS is not equal to zero (meaning that there was, indeed, valid data in the data file), then the following events take place:

- NEWDATA is sorted by the application's sort key sequence. The sort key sequence in the BY statement was left blank in the example in **Appendix A**, because it is dependent upon the sort key sequence for your particular application. In this particular application, observations with duplicate key variable values are considered duplicates and are not wanted. So, they are removed via the NODUPKEY sort option. Your own applications may not need this particular sort option.

- The production SAS data set is allocated with a LIBNAME of PRODFILE. The permanent SAS data library used in the LIBNAME statement is input via the &PRODFILE SAS Macro parameter passed to the READFLAT SAS Macro.

- The PRODFILE.PRODDATA permanent SAS data set is updated with the NEWDATA SAS data set via the MODIFY statement with the KEY option. This is possible because PRODFILE.PRODDATA is an indexed SAS data set. The variable specified in the KEY option is the same one used to sort NEWDATA. An attempt is made to match each observation in NEWDATA with one containing the same key variable value in PRODFILE.PRODDATA. The _IORC_ automatic variable is used to determine whether or not a match is found. When a match is found, some updating of the variables in the observation read from PRODFILE.PRODATA is done ("…*Other SAS Statements*…") and the observation is replaced. Otherwise the observation is added to PRODFILE.PRODDATA.

If NUMBROBS *is* equal to zero (meaning that there was no valid data in the data file), then a message is written to the SAS Log. No attempt is made to update the permanent SAS data set.

Whether there is valid data in NEWDATA or not, the READFLAT SAS Macro deletes the now processed data file via the FDELETE function. This is necessary so that the data file is not processed again when this application is run in the future. It promotes good housecleaning, by removing obsolete files from the data directory.

The final act of the READFLAT SAS Macro is to deallocate the FILEREF of the data file. This ensures that there is no conflict for the "DATAFIL" FILEREF when multiple executions of the READFLAT SAS Macro take place.

Conclusions

The three SAS programs discussed in this paper offer a simple, seamless system that you can use to process a varying number of new data files from a data directory. You can easily copy them to your own

system, and modify them for use with one of your own applications. This could save you a lot of development and maintenance time. Why not give this a try?

Disclaimer

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

References

Raithel, Michael A. 2003. *Tuning SAS® Applications in the OS/390 and z/OS Environments, Second Edition.* Cary, NC: SAS Institute Inc.

Contact Information

Please feel free to contact me if you have any questions or comments about this paper. You may reach me at:

Michael A. Raithel
Westat
1650 Research Boulevard
Room RW4521
Rockville, Maryland 20850

michaelraithel@westat.com

Appendix A. - SAS Programs

This appendix contains the three SAS programs used for processing all data files found in a specified directory.

Driver_Program.sas

```
*************************************************************************;
* Program: Driver_Program.sas                                           *;
*                                                                       *;
* Purpose: This program %INCLUDEs several SAS Macro programs and then   *;
*          executes them so that data files in specified directories may be *;
*          processed.                                                   *;
*************************************************************************;

*****************************************************************;
* SAS Options, Formats, etc.                                    *;
*****************************************************************;
options noxwait xmin xsync;

*************************************************************************;
* INCLUDE the Process_Flat_Files.sas SAS program that contains the      *;
* READFLAT SAS Macro. That Macro reads an individual data file's records*;
* and stores the data in a permanent SAS data set.                      *;
*************************************************************************;
%include 'Q:\programs\Process_Flat_Files.sas';

*************************************************************************;
* INCLUDE the SAS Macro program that identifies individual data files   *;
* in the specified data directory and builds macro calls that, in turn, *;
* execute the READFLAT SAS Macro in  the Process_Flat_Files.sas SAS     *;
* program for each file.                                                *;
*************************************************************************;
%include 'Q:\programs\Identify_Flat_Files.sas';

*************************************************************************;
* Execute the IDENTIFY SAS Macro in the Identify_Flat_Files.sas program  *;
* to identify and process the data files in each specified data directory*;
*************************************************************************;

%IDENTIFY(Q:\data,\\system1\input\datadir);

%IDENTIFY(Q:\data,\\system2\input\datadir);

%IDENTIFY(Q:\data,\\system3\input\datadir);
```

Identify_Flat_Files.sas

```
*******************************************************************************;
* Program: Identify_Flat_Files.sas                                            *;
*                                                                             *;
* Purpose: This program reads all of the data file names in a directory and   *;
*          creates calls to the READFLAT SAS Macro, located in the            *;
*          Process_Flat_Files.sas program.                                    *;
*                                                                             *;
*          This Macro program accepts two parameters:                         *;
*                                                                             *;
*            SASLIB - This is the full path name of the production SAS data   *;
*                     library. This value will be coded into the call for the*;
*                     READFLAT SAS macro, that is the output of this program *;
*            DIRNAME - The full path name of the directory that contains the  *;
*                      data files.                                            *;
*                                                                             *;
*            NOTE: This program expects that all data files will have an      *;
*                  extension of '.txt'!                                       *;
*******************************************************************************;

  /*************************************************/
  /* Beginning of the IDENTIFY SAS Macro.          */
  /*************************************************/
%MACRO IDENTIFY(SASLIB,DIRNAME);

*********************************************************************;
* Pipe the names of data files to the SAS System for processing.    *;
*********************************************************************;
filename dircmd pipe "dir /b &DIRNAME\*.txt";

*********************************************************************;
* Temporary file to hold the READFLAT SAS Macro call statements.    *;
*********************************************************************;
filename holdmacs TEMP;

*********************************************************************;
* Process each specific file name, dropping unneeded ones. Format   *;
* SAS READFLAT Macro calls for valid data files.                    *;
*********************************************************************;
data _null_;

length outline $100;

file holdmacs;

infile dircmd missover length=length;

input @;

input bigline $varying200. length;

outline = '%READFLAT(' || "&SASLIB" || ',' || "&DIRNAME" || '\' || trim(left(bigline)) || ');'
;

put outline;

run;
```

```
*********************************************************************;
* Include holdmacs, which is now a series of READFLAT SAS Macro     *;
* invocations. This will execute Process_Flat_Files.sas once for    *;
* each ".txt" flat file found in the target data directory.         *;
*********************************************************************;
%INCLUDE holdmacs;


  /*****************************************************/
  /* End of the IDENTIFY SAS Macro.                    */
  /*****************************************************/
%MEND IDENTIFY;
```

Process_Flat_Files.sas

```
*****************************************************************************;
* Program: Process_Flat_Files.sas                                           *;
*                                                                           *;
* Purpose: This program processes each data file and updates the production SAS *;
*          data set.                                                        *;
*                                                                           *;
*          This program is in the form of a SAS Macro and accepts two       *;
*          parameters:                                                      *;
*                                                                           *;
*            PRODFILE - This is the full path filename of the SAS data library  *;
*                       where the production SAS data set used to store the     *;
*                       processed data files resides.                       *;
*            DATAFIL - This is the full path filename of the data file that     *;
*                       will be processed by this program.                  *;
*****************************************************************************;

%MACRO READFLAT(PRODFILE,DATAFIL);
*********************************************************************;
* Allocate the Data file.                                           *;
*********************************************************************;
filename DATAFIL "&DATAFIL";

*********************************************************************;
* Determine if the Data file is in use.                             *;
*********************************************************************;
data _null_;

inuse = fopen('DATAFIL');

call symput('INUSE',inuse);

if inuse = 0 then do;

       put '*** Attention: the file below was in use ***';
       put '*** and could not be processed           ***';

       put "&DATAFIL";
       put _all_;
       put '*** Attention: the file above was in use ***';
       put '*** and could not be processed           ***';

end;

run;
```

```
%IF &INUSE NE 0 %THEN %DO;
*********************************************************************;
* Process the records in the data file.                             *;
*********************************************************************;
data newdata;

… SAS code to extract the data from the data file…

run;

*********************************************************;
* Determine if there are any obs in NEWDATA.            *;
*********************************************************;
proc sql noprint;
select nobs - delobs into :numbrobs
       from dictionary.tables
        where libname = "WORK" and
              memname = "NEWDATA"
              ;
quit;

  /*******************************************************/
  /* Only do the following if there are obs in NEWDATA. */
  /*******************************************************/
%IF &NUMBROBS NE 0 %THEN %DO;

*********************************************************************;
* Sort the data.                                                    *;
*********************************************************************;
proc sort data=newdata nodupkey;
        by …;
run;

*********************************************************************;
* Allocate the production SAS data set.                             *;
*********************************************************************;
libname  prodfile "&PRODFILE";

*************************************************************;
* Update the production SAS data set.                       *;
*************************************************************;
data prodfile.proddata;

set newdata;

modify prodfile.proddata key=…;

select(_iorc_);

        when(%sysrc(_sok)) do; /* A match the key */
                …Other SAS Statements…
                replace;
        end;

        when (%sysrc(_dsenom)) do; /* Not a match on the key */
                _error_ = 0;
                output;
        end;

        otherwise;
end;

run;
```

```
*******************************************************************;
* Deallocate the production SAS data set.                         *;
*******************************************************************;
libname prodfile clear;


  /*********************************************************/
  /* End of DO stmt. to process obs in NEWDATA.          */
  /*********************************************************/
%END;



%ELSE %DO;
*******************************************************************;
* Make note in log that no valid data were found in the data file. *;
*******************************************************************;
data _null_;

       put '*** Attention: No data records were found in this file ***';
       put "Data File: &DATAFIL";
       put '*** Attention: No data records were found in this file ***';

run;

  /*********************************************************/
  /* End of DO stmt. to flag no data records found in file*/
  /*********************************************************/
%END;

*******************************************************************;
* Delete the data file.                                           *;
*******************************************************************;
data _null_;

       rc = fdelete('DATAFIL');

       put '*****************************************';
       put 'FILE DELETE RETURN CODE = ' rc;
       put '*****************************************';

run;

  /*********************************************************/
  /* End of DO stmt. to process data files not in use.   */
  /*********************************************************/
%END;

*******************************************************************;
* Deallocate the data file fileref.                               *;
*******************************************************************;
filename DATAFIL clear;

%MEND READFLAT;
```