SAS® GLOBAL FORUM 2018

USERS PROGRAM

Files Arriving at an Inconvenient Time? Let SAS® Process Your Files with FILEEXIST While You Sleep

Matthew Shotts

Educational Testing Service

April 8 – 11 | Denver, CO

#SASGF

Files Arriving at an Inconvenient Time? Let SAS® Process Your Files with FILEEXIST While You Sleep

Matthew Shotts

Educational Testing Service

ABSTRACT

The FILEEXIST and SLEEP functions can be paired together to iteratively scan a location on a network drive for the arrival of a file. This paper provides a simple framework to control the interval between these attempts to locate the file and also control the acceptable number of scans to perform before ceasing the operation. These controls are accomplished through parameters defined in %LET statements and executed through a %DO %UNTIL processing loop. The example provided can be a stepping stone for beginner or intermediate programmers to understand the basics of macro processing and do loops while also providing them with a helpful tool to automate their file processing.

METHOD (PROCESS DETAILS)

1. Place the example code in front of the code you would typically use to process a file.

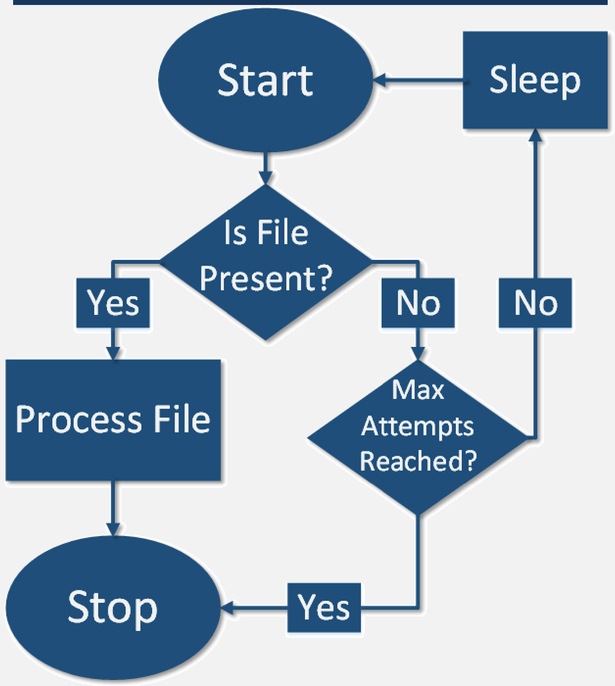
2. START: The %do %until conditions are not met since the macros &filehere and &count are set to 'N' and 0 (respectively) on initial execution. SAS moves to the fileexist step and searches for the file.

3. STOP: If the file is found then &filehere is updated from 'N' to 'Y' which closes out the %do %until loop. SAS will then progress on to the code you typically use to process the file.

3. If the file is not found then &count has +1 added, SAS sleeps for the specified &Interval, and then the loop resumes at the beginning.

4. STOP: If the loop runs enough times to make &count match the specified &MaxAttempt then a warning is posted to the log and operations are cancelled.

METHOD (PROCESS FLOW)



METHOD (EXAMPLE CODE)

```
%let MaxAttempt=12; /*Define the number of desired search attempts                                  */
%let Interval=5;    /*Define Interval (seconds) between attempts                                    */
%let filehere=N;/*DON'T CHANGE, Remains 'N' until the file is found after which it is switched to 'Y'  */
%let count=0;   /*DON'T CHANGE, Set to 0 and +1 is added after each attempt to locate the file     */

%macro find;
%do %until ("&filehere."="Y" or &count.=&MaxAttempt.); /* These are the criteria for the loop to close  */

%if %sysfunc(fileexist("C:\Users\mshotts\Documents\example_input.dat"))
    %then %do; /* If file is found then the &filehere is updated to Y to close the %do %until loop  */
        %let filehere=Y;
    %end;

    %else %do; /* If file is not found then SAS goes to sleep and the &count of the attempts has 1 added  */
        %let count=%eval(&count.+1); /* &count is the number of search attempts for the file          */

        data _null_;
            slept=sleep(&Interval.);
        run;
    %end;
%end;

%if &count.=&MaxAttempt. %then %do; /* Ends process if file is not found after specified number of attempts */
    %let waittime=%eval((&MaxAttempt.*&Interval.)/60); /* This transforms the search time into minutes   */
    %put WARNING: The file has not yet appeared after &waittime. minute(s), Contact your data provider;
    %abort cancel; /* %abort cancel ensures that SAS stops here after reaching the max attempt threshold  */
%end;

%mend find;
%find;
/* Insert the code for processing the file here, SAS will proceed to this line once the file is found   */
```

Files Arriving at an Inconvenient Time? Let SAS® Process Your Files with FILEEXIST While You Sleep

Matthew Shotts

Educational Testing Service

Description of the functions used in example code

FILEEXIST

The Fileexist procedure can be used to determine if a file is present at a given location. In the example, fileexist is used as a gatekeeper. If file is present, the gate is opened and SAS processes the file. If not present, the gate remains closed.

SLEEP

The Sleep function can be used to make SAS pause processing for a specified interval. Please note that SAS defaults to seconds as the unit of time. This can easily be adjusted to minutes or hours by multiplying against the &Interval in the sleep statement (i.e. slept=sleep(&Interval.*60); would turn the sleep interval into minutes).

In the example, sleep is used to create a period of inactivity in between each attempt to locate the file.

%DO %UNTIL

The %do %until function can be used to create a loop of various actions which conclude once a criteria is met. For the example here, there are two criteria for closing the loop: either the file is found via fileexist or the number of attempts defined in &MaxAttempts is reached. %Do %Until must be enclosed within a macro in order to function. The %macro *find* is filling this role in the example. All %do commands also need to be closed out with a corresponding %end.

%IF %THEN %DO and %ELSE %DO

Conditional processing such as if-then do isn't just for the data step. You can also take advantage of %if %then %do to conditionally execute whole data steps and procs. As with %do %until above, these must be enclosed within a macro and any %do commands closed out with an %end. In the example, if the file is present then the &filehere is updated to 'Y' to close out the loop. If not present, the &count of the attempts has +1 added and SAS goes to sleep.

This is followed by another %if %then %do to determine if the maximum number of attempts has been reached. If so, SAS aborts processing but remains open through the use of %abort cancel.

METHOD (EXAMPLE CODE)

```
%let MaxAttempt=12; /*Define the number of desired search attempts                                    */
%let Interval=5;    /*Define Interval (seconds) between attempts                                     */
%let filehere=N;/*DON'T CHANGE, Remains 'N' until the file is found after which it is switched to 'Y'*/
%let count=0;   /*DON'T CHANGE, Set to 0 and +1 is added after each attempt to locate the file     */

%macro find;
%do %until ("&filehere."="Y" or &count.=&MaxAttempt.); /* These are the criteria for the loop to close  */

%if %sysfunc(fileexist("C:\Users\mshotts\Documents\example_input.dat"))
    %then %do; /* If file is found then the &filehere is updated to Y to close the %do %until loop  */
        %let filehere=Y;
    %end;

    %else %do; /* If file is not found then SAS goes to sleep and the &count of the attempts has 1 added */
        %let count=%eval(&count.+1); /* &count is the number of search attempts for the file          */

        data _null_;
            slept=sleep(&Interval.);
        run;
    %end;
%end;

%if &count.=&MaxAttempt. %then %do; /* Ends process if file is not found after specified number of attempts */
    %let waittime=%eval((&MaxAttempt.*&Interval.)/60); /* This transforms the search time into minutes  */
    %put WARNING: The file has not yet appeared after &waittime. minute(s), Contact your data provider;
    %abort cancel; /* %abort cancel ensures that SAS stops here after reaching the max attempt threshold */
%end;

%mend find;
%find;
/* Insert the code for processing the file here, SAS will proceed to this line once the file is found  */
```

SAS® GLOBAL FORUM 2018

April 8 – 11 | Denver, CO
Colorado Convention Center

Presenter Contact Information:
Matthew Shotts
mshotts@ets.org

#SASGF

Paper 2562-2018

Files Arriving at an Inconvenient Time? Let SAS® Process Your Files with FILEEXIST While You Sleep

Matthew Shotts, Educational Testing Service

ABSTRACT

The FILEEXIST and SLEEP functions can be paired together to iteratively scan a location on a network drive for the arrival of a file. This paper provides a simple framework to control the interval between these attempts to locate the file and also control the acceptable number of scans to perform before ceasing the operation. These controls are accomplished through parameters defined in %LET statements and executed through a %DO %UNTIL processing loop. The example provided can be a stepping stone for beginner or intermediate programmers to understand the basics of macro processing and do loops while also providing them with a helpful tool to automate their file processing.

INTRODUCTION

Do you work with international clients who send data to you at 4 in the morning? Or perhaps you often find yourself double-booked with your responsibilities for processing data and attending meetings or other project coordination activities. Wouldn't it be nice to have a way to have SAS process your data files once they became available so you didn't have to monitor the network drive yourself? Fortunately for you, this is possible and this paper will provide and describe code which can iteratively check the network drive for the arrival of your input file.

PROCESS FLOW

Before moving on to the code, let's first define the process. The goal is to have SAS search the network drive for the presence of an input file. If the file is found, then SAS should process it. If the file is not found then SAS should ideally pause for a specified interval of time before trying again. However, SAS should not continue this loop of checking and pausing into perpetuity so it would be good to have a control that limits the number of attempts. The ideal process flow might look like the following in Figure 1.

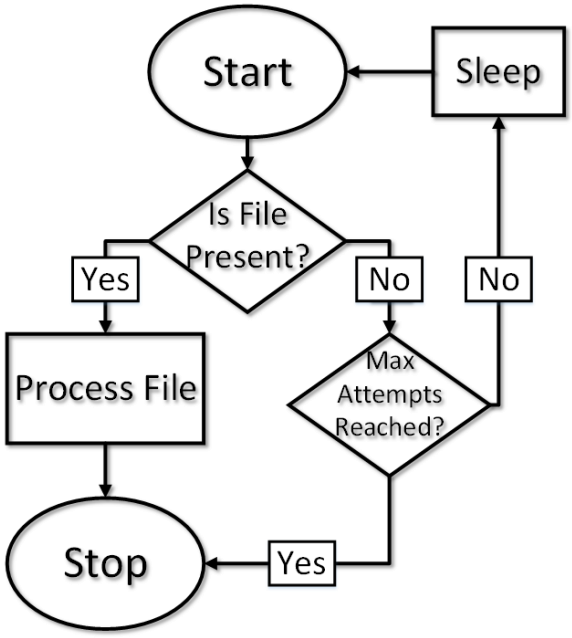


Figure 1. Process Flow

CODING THE PROCESS FLOW

There are a number of ways to code the process flow from Figure 1 but functions like FILEEXIST, SLEEP, %IF %THEN %DO, and %DO %UNTIL seem particularly well-suited to the task.

FILEEXIST

The FILEEXIST function can be used to determine if an external file is present or not. The syntax is simply FILEEXIST(*location and name of file*). FILEEXIST can be used with macro conditional processing if paired with the %SYSFUNC function like the following:

```
%sysfunc(fileexist("location and name of file"))
```

SLEEP

The SLEEP function can be used to halt SAS processing for a specified interval of time. The syntax for the function is SLEEP(*n*) where *n* is the specified interval as measured in seconds. The interval can be changed to minutes by multiplying *n* by 60 within the parenthesis. SAS encourages users to call the sleep function within a null data step like the following:

```
data _null_;
  slept= sleep(30);
run;
```

%IF %THEN %DO AND %ELSE %DO

Most SAS users are probably familiar with using if-then conditional logic to manipulate data within a data step. What some may be less familiar with is using this same if-then logic at the macro level. The macro %IF %THEN %DO function can be used to conditionally execute whole data steps and procs. For example, if the file is present then SAS can be directed to process the file. If the file is not present, then SAS can be directed to go to sleep:

```
%if %sysfunc(fileexist("location and name of file")) %then %do;
      /*Code for processing the file if it is present*/
      %end;

%else %do;
      data _null_;
            slept= sleep(30);
      run;
      %end;
```

Please note that the prerequisite for using conditional processing with macros is that it must occur within the boundaries of a macro statement. Additionally, all %DO commands must be closed out with a corresponding %END.

%DO %UNTIL

The %DO %UNTIL function can be used to create a loop of various activities that only conclude once a specific condition has been met. The syntax for the %DO %UNTIL function is %do %until (*condition to close the loop*); For example, the following condition could be set to close out the FILEEXIST loop:

```
%do %until (%sysfunc(fileexist("location and name of file")));
```

Same as the %IF %THEN %DO function, %DO %UNTIL will only execute within a macro statement and must be closed out with a corresponding %end.

PUTTING IT ALL TOGETHER

Now that all the various functions have been introduced, let's go over how to make the process flow a reality. The following SAS code can be copied and pasted in front of any code which processes an input file and will hold off on executing that particular code until the input file is present.

1. The code begins by allowing the user to define the maximum number of desired attempts to search for the file and also define the interval in seconds between each of those attempts. Two macro variables, filehere and count, are also set to their baseline values. Both of these help control when the %DO %UNTIL loop is exited and should not be changed.
2. As mentioned earlier, all conditional processing with macros must happen within a macro statement so the macro statement, find, is created.
3. %DO %UNTIL comes next since SAS now needs to execute a loop of activities. We also defined the criteria for exiting the loop which is the &filehere macro being equal to "Y" or if the &count macro variable matches the &MaxAttempt macro variable defined by the user.
4. Fileexist is called to determine if the file is present. If it is, then the &filehere macro variable is updated from "N" to "Y" and the loop will close.
5. If the file is not found, then the &count macro variable has +1 added and sleep is triggered for the length of time defined in the &Interval macro variable. The loop then resumes at the top once the sleep is complete.
6. If the loop continues long enough that the &count and &MaxAttempts macro variables match then the loop is exited. A message is written to the log summarizing the time spent and an %abort cancel is triggered which cancels any remaining processing in the SAS program.
7. The code for processing the input file should start after this point.

```
/* See bullet 1                                                    */
%let MaxAttempt=12; /* Define the number of desired search attempts */
%let Interval=5;    /* Define Interval (seconds) between attempts  */
%let filehere=N;    /* DON'T CHANGE, Remains 'N' until the file is found
                       after which it is switched to 'Y'           */
%let count=0;       /* DON'T CHANGE, Set to 0 and +1 is added after each
                       attempt to locate the file                  */
/* See bullet 2                                                    */
%macro find;
/* See bullet 3, These are the criteria for the loop to close      */
%do %until ("&filehere."="Y" or &count.=&MaxAttempt.);

/* See bullet 4                                                    */
%if %sysfunc(fileexist("C:\Users\mshotts\Documents\example_input.dat"))
/* If file is found then &filehere is updated to Y to close the %do %until
      loop                                                         */
      %then %do;
            %let filehere=Y;
      %end;

/* See bullet 5, If file is not found then SAS sleeps and the &count of the
      attempts has +1 added                                        */
      %else %do;
/* &count is the number of search attempts for the file            */
            %let count=%eval(&count.+1);
            data _null_;
                  slept=sleep(&Interval.);
            run;
```

```
      %end;
%end;

/* See bullet 6, Ends processing if file is not found after specified number
      of attempts                                                          */
%if &count.=&MaxAttempt. %then %do;
/* This transforms the search time into minutes and posts to the log      */
      %let waittime=%eval((&MaxAttempt.*&Interval.)/60);
      %put WARNING: The file has not yet appeared after &waittime. minute(s),
      Contact your data provider;
/* %abort cancel stops SAS processing once the max attempt threshold is hit*/
      %abort cancel;
%end;

%mend find;
%find;
/* See bullet 7, Insert the code for processing the file here, SAS will
      proceed to this line once the file is found                          */
```

HELPFUL TIPS

The following are some helpful tips that should help you get the most out of the code:

1. The use of %LET macro variables for the user to define the location and name of the input file is recommended. If this change is made then be sure to update the FILEEXIST to refer to the %LET input macro variable. If %LET is not used then the user will have to update the FILEEXIST function with the new input file's location and name within the code itself before each run.
2. If the loop is running and you need to prematurely exit, pressing Ctrl+Break will bring up the SAS Tasking Manager and allow you to Cancel Submitted Statements.
3. If there are a number of SAS programs which would benefit from this code, then it would be advisable to save this as a stand-alone SAS program and call that program in as needed using a %INCLUDE statement.
4. Care should be taken when defining the &MaxAttempt macro variable as each iteration of the loop adds more information to the SAS log. If the SAS log reaches capacity then the user will be prompted to address the situation and SAS will stop processing while waiting for the user's input. If reaching capacity seems a likely scenario given your needs, then you can expand the number of rows permitted in the log using the DMSLOGSIZE option or you could change the options so that the log is suppressed while the %DO %UNTIL loop is active.
5. The provided code only writes a warning to the log if the file is not found during the expected time frame. Additional actions such as documenting the incident in an external file or sending an email to the appropriate contacts could be added.

CONCLUSION

The code shared in this paper can serve you in a number of situations where it may not be feasible to be present to schedule your SAS program. The macro functions demonstrated are also particularly powerful and can be adopted for a wide variety of other uses. Armed with this knowledge you can be productive all the time; even when you sleep.

RECOMMENDED READING

Fehd, R. J. 2007. "Do Which? Loop, Until or While? A Review of Data Step and Macro Algorithms." *Proceedings of the SAS Global Forum 2007 Conference*, Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2007/067-2007.pdf

Libeg, L. 2007. “Functionsize: Process Your External Files.” *Proceedings of the SAS Global Forum 2007 Conference*, Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2007/054-2007.pdf.

SAS Support. “FILEEXIST Function”. Accessed March 6, 2018. http://documentation.sas.com/?docsetId=lefunctionsref&docsetTarget=n06xm8hwk0t0axn10gj16lfiri43.htm&docsetVersion=9.4&locale=en.

SAS Support. “SLEEP Function: Windows”. Accessed March 6, 2018. http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-func-sleep.htm.

CONTACT INFORMATION

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

Matthew Shotts
Educational Testing Service
Princeton, NJ 08016
E-mail: mshotts@ets.org