Paper CC12

RUN; RUN; RUN; - Methods for Running Multiple Programs in a Series

Robert Matthews, University of Alabama at Birmingham

ABSTRACT

Anyone who has ever had to run a series of programs multiple times in a row has probably thought about ways to automate the process. For example, if you have 25 programs that need to be run one after another, the normal method would be to run the first program, wait for it to finish, then submit the next one, and so on. If you have the ability to run multiple SAS® sessions, then you can speed up the process a bit by submitting programs in each session. However, it still takes some time and effort to monitor the programs, wait for each one to finish, and then submit the next program in the series. We encountered this issue several years ago and have developed two methods for implementing a “hands-off” approach for submitting a series of programs. Some additional features we implemented include the ability to either stop or continue processing the remaining programs if an individual program in the series encounters an error as well as the ability to send email messages after individual programs in the series have been run. These methods greatly reduce the need for manual intervention when running a long series of programs and help alleviate an otherwise laborious and potentially error-prone process.

INTRODUCTION

We regularly receive a set of Medicare data files known as the Chronic Condition Warehouse (CCW) with claims on a sample of beneficiaries from across the nation. The set of files that we receive for each year of data has approximately 25 files/year and we sometimes receive 6 or 7 years of data in each shipment. This results in over one hundred separate programs that need to be run in order to read in all the data. To help alleviate possible errors during the production phase of reading in large amounts of data from many different files, we came up with two methods that help reduce the burden for the programmer or analyst who is tasked with running hundreds of individual programs.

Methods for running multiple programs:

1. Batch file to run multiple programs and pass parameters to a SAS® program (see Appendix I for example)
2. “Master” program to “call” or include each individual program (see Appendix II for example)

METHOD I

This method is useful if you want to have individual log and output files created for each program in the series. However, you can also concatenate the individual files into a single file at the conclusion of the script so that you end up with a single log and a single output file. This can be advantageous for several reasons; for example, when checking for errors, it is much easier to search one file instead of dozens or hundreds of individual files.

One of the key components of using this method is to be able to define one or more variables for the execution of each individual program and have these variables defined in the batch file so that each individual program doesn’t have to be modified. In the example in Appendix I, this variable is named YEAR. In the batch file, the year is passed to the SAS® program on the command line where SAS® is invoked. Once the program begins execution, the first step should be to retrieve this environment variable and place it into a global SAS® macro variable so that it can be used throughout the rest of the program. An example of how this works in practice is given below.

A DOS environment variable named YEAR is defined in the batch file (see Appendix I) for each execution of the program. This variable is then turned into a normal SAS® macro variable using a DATA step like the following:

```
Data _null_;
   length yr $4.;
   yr = sysget("YEAR"); * extract data from OS environment variable;
   %global SAS_YEAR;
   call symputx("SAS_YEAR", yr);  * create a SAS macro variable;
run;
```

METHOD II

One of the primary advantages of using this method is that all the programs are run within the SAS environment. This makes it easier to create macro variables and to utilize other SAS components such as "progress bars". This method is also useful if you want to have a single log and output file created for the entire series of programs. This can be advantageous for several reasons; for example, when checking for errors, it is much easier to search one file than dozens or hundreds of individual files.

The SOURCE2 option on the %include statements allow you to display the source code of each included program. This option can be removed if desired.

The example program shows some other features, such as the ability to print a date/time stamp at the beginning and end of the program execution and how to begin program execution at a certain point in time (for example, in the middle of the night). The latter is accomplished through the use of the SLEEP function.

A block of code (see Appendix III) could also be used anywhere in the "master" program to indicate the status of the program series. This would send an email message to a designated user letting them know what program within the series has just finished running. In this example, an email message is sent to a specific user after each year of data is processed.

CONCLUSION

Each of the methods described above have both advantages and disadvantages. Method I has the advantage of creating separate log and output files for each program. In addition, this method uses less memory than Method II because there is no overhead for the SAS® Display Manager windows. However, the disadvantage is that it is operating system (OS) dependent because it requires a script file to run the SAS programs in batch mode and utilizes OS commands to rename, delete, and concatenate files.

Method II has the advantage of having a single log and output file, and allows for a "progress bar" to show the progress of individual programs. In addition, this method makes it easy to pass SAS® macro variables into each of the individual programs and keeps everything within the SAS® environment.

When set up and tested properly, both of these methods allow for "hands-free" operation in running a long series of programs. Email alerts can be configured and sent during key stages of the program series and a "progress bar" can be implemented to inform the user about the status of each stage. Finally, these methods greatly reduce the need for manual intervention when running a long series of programs and help alleviate an otherwise laborious and potentially error-prone process.

CONTACT INFORMATION

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

Robert Matthews

University of Alabama at Birmingham

510 20th St. S., FOT 805D

Birmingham, AL 35294

rsm@uab.edu

APPENDIX I

Batch file method for submitting multiple SAS® programs in a Microsoft Windows environment

Each line below would be a separate line in a single batch file (the file extension could be either .BAT or .CMD)

Explanation of SAS command line options:

- The –CONFIG option specifies the SAS® configuration file that is used by SAS
- The –SYSIN option specifies the input SAS® program that will be executed
- The –SET option specifies an operating system environment variable that will be passed to the SAS® program

BATCH FILE BEGINS ON THE FOLLOWING LINE

```
set YEAR=2011
"C:\Program Files\SASHome\x86\SASFoundation\9.3\sas.exe" -CONFIG "N:\S1\sasv9.cfg"
-SYSIN "H:\studies\pgms\test.sas" -SET YEAR "%YEAR%" [this is part of the line above]
del test_%year%.log
del test_%year%.lst
rename "test.log"="test_%year%.log"
rename "test.lst"="test_%year%.lst"

set YEAR=2010
"C:\Program Files\SASHome\x86\SASFoundation\9.3\sas.exe" -CONFIG "N:\S1\sasv9.cfg"
-SYSIN "H:\studies\pgms\test.sas" -SET YEAR "%YEAR%" [this is part of the line above]
del test_%year%.log
del test_%year%.lst
rename test.log=test_%year%.log
rename test.lst=test_%year%.lst

set YEAR=2009
"C:\Program Files\SASHome\x86\SASFoundation\9.3\sas.exe" -CONFIG "N:\S1\sasv9.cfg"
-SYSIN "H:\studies\pgms\test.sas" -SET YEAR "%YEAR%" [this is part of the line above]
del test_%year%.log
del test_%year%.lst
rename test.log=test_%year%.log
rename test.lst=test_%year%.lst

set YEAR=2008
"C:\Program Files\SASHome\x86\SASFoundation\9.3\sas.exe" -CONFIG "N:\S1\sasv9.cfg"
-SYSIN "H:\studies\pgms\test.sas" -SET YEAR "%YEAR%" [this is part of the line above]
del test_%year%.log
del test_%year%.lst
rename test.log=test_%year%.log
rename test.lst=test_%year%.lst

rem Concatenate all the individual LOG and LST files back into single LOG & LST files
copy test_2011.log + test_2010.log + test_2009.log + test_2008.log   test.log
copy test_2011.lst + test_2010.lst + test_2009.lst + test_2008.lst   test.lst
```

APPENDIX II

```
/*data; rc=sleep(10000); run; * this step is useful if you want the macro below to
start after a specified number of seconds.*/

%put START TIME: %sysfunc(datetime(),datetime14.);  * print a date/time stamp in the
log when the program started executing;

%macro read;
       %do year=2007 %to 2012;
              * create a 2-digit year variable used in each individual program;
              %let year2=%substr(&year.,3,2);

              * used in individual programs to specify the infile path;
              %let file_path=O:\PV\Request_4187\&year.\;

              * used in the %include lines below to specify the program path;
              %let pgm_path=H:\studies\PV\pgms\&year.\;

              * Define a "progress bar" window to show the user what year of data is
                currently being processed;
              %window progress irow=4 rows=7 columns=40 #1 @6 "Processing YEAR: &year"
                      persist=yes;
              %display progress NOINPUT;

              %inc "&pgm_path.mbsf_ab_summary_read_v8-v2.sas" / source2;
              %inc "&pgm_path.mbsf_d_cmpnts_read_v8-v2.sas" / source2;
              %inc "&pgm_path.dme_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.dme_line_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.hha_base_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.hha_revenue_center_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.hospice_base_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.hospice_revenue_center_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.inpatient_base_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.inpatient_revenue_center_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.outpatient_base_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.outpatient_revenue_center_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.snf_base_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.snf_revenue_center_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.bcarrier_line_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.bcarrier_claims_j_read_v8-v2.sas" / source2;
              %inc "&pgm_path.pde_saf_file_read_v8-v2.sas" / source2;

              %inc "code in Appendix III to send an email to a user indicating the
                    program status";
       %end;
%mend;

%read;

%put END TIME: %sysfunc(datetime(),datetime14.); * print a date/time stamp in the log
                                                   when the program finishing running;
```

APPENDIX III

```
%let pgm_path=H:\studies\PV\pgms\Robert\;
%let pgm_name=TEST program.sas;

options emailhost="your.email.server" emailid="rsmm@uab.edu" emailsys="SMTP";
filename outgoing email  to="rsmm@uab.edu"
                    subject="&pgm_name. has finished running for year: &year";

data _null_;
      file outgoing;
      put "Program name: &pgm_name. " /;
      put 'Please review LOG file for any errors before using dataset.';
      put "Log filename: &pgm_path.&pgm_name._log.txt";
      put // "Test of sending an email from within a running SAS program.";
run;
```