Paper T05-2007

# Using SAS® to Control and Automate a Multi SAS Program Process

Patrick Halpin, dunnhumby USA, Cincinnati, OH

## ABSTRACT

Often times a project is comprised of many SAS® programs that need to run in a given order. Additionally, the SAS programs are dependent on previous SAS programs completing. One way to accomplish and automate this task is to use "Done" files and the SLEEP command in SAS. "Done" files are files that are created when a SAS program finishes. SAS programs dependent on previous jobs finishing look for those "Done" files to be created. Once the "Done" files are created the next SAS job in the process will start.

## INTRODUCTION

How many times has one sat around waiting for a program to end to kick off the next one? Just batch the second to run later. What if you don't know when it will finish? Create a shell script that runs it after the first. What if you have multiple dependent programs? Buy scheduling software. What if it is just too expensive? Then read this!

When a project consists of multiple SAS programs and they need to run in a given order the use of a UNIX shell script, "Done" files and the SLEEP Command can be very useful tools to not only accomplish this task but also help in the automation of it.

- The UNIX shell script runs the SAS programs in the correct order.
- The "Done" files can be files or SAS data sets that are created when a SAS program finishes.
- The SLEEP Command is used in SAS programs that are dependent on previous SAS jobs finishing.

This paper will go into detail on how to accomplish this task step by step:

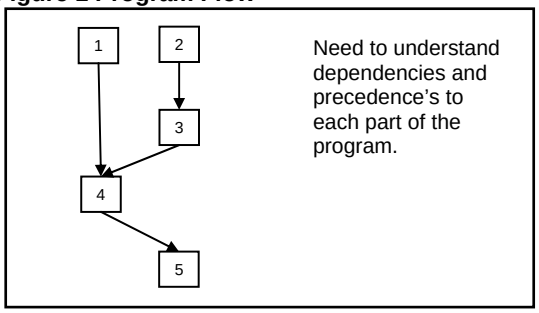
1) Creating the UNIX control Shell Script
2) Creating "Done" files
3) Using the SLEEP command
4) Putting it all together

## STEP 1 – CREATING UNIX SHELL SCRIPTS

The first thing you will need to do is list out the SAS programs in the order they need to run. You have to keep in mind what programs are dependent on other programs and which programs can be run at the same time. In the example below there will be 5 SAS programs in the process:

- Prog_1 and prog_2 can be run at the same time
- Prog_3 needs prog_2 to finish
- Prog_4 needs prog_1 and prog_3 to finish
- Prog_5 needs prog_4 to finish

**Figure 1 Program Flow**




The UNIX shell script below is written in the Korn shell (program name batching.csh).

```
#!/bin/ksh
cd /Program_Directory/
sas prog_1.sas &
```

```
sas prog_2.sas
sas prog_3.sas
sas prog_4.sas
sas prog_5.sas
```

The first line of the shell script tells UNIX what language the program is written in. The second line changes to the directory the programs are in. The third line starts the first program in the process running. The "&" after prog_1.sas allows prog_1.sas and prog_2.sas to run at the same time. When prog_2.sas is complete prog_3.sas runs. After prog_3.sas completes prog_4.sas runs and after completion prog_5.sas runs. The last point is **NOT** what we want to do; we want prog_4.sas to start after **BOTH** prog_3.sas **AND** prog_1.sas. This is where the "DONE" files and SLEEP command come in.

## STEP 2 – "DONE" FILES

The second piece of the puzzle is creating "Done" files. A "Done" file is a file that is created when a SAS program has finished running. For this paper the "Done" file will actually be a SAS data set (the appendix shows how to use text files as "Done" files). Creating these "Done" files is easy. Two steps need to be added to a SAS program. At the start of the program you need to delete the "Done" file if it already exists.

```
Libname lib_done "/program_directory/";
%macro delete_done(var1);
%if  %sysfunc(exist(lib_done.done_file&var1)) %then %do;
   proc datasets lib=lib_done;
      delete done_file&var1;
   run;
%end;
%mend delete_done;
%delete_done(program_name);
run;
```

The delete_done macro checks for an existing done file by using the exist function and program name parameter passed to the macro. If the data set exists then it will be deleted. At the end of the program you will need to create the "Done" file

```
%macro create_done(var1);
Data lib_done.done_file&var1;
   done="YES";
   output;
run;
%mend create_done;
%create_done(program_name);
run;
```

The create_done macro creates a SAS dataset out in the done library via a simple data step. All one needs to do is pass the macro the program name.

## STEP3 – THE SLEEP COMMAND

The SLEEP command allows one to pause the current SAS program. This is often done when one is opening and using SAS to control EXCEL. In that case one will usually let SAS "sleep" a few seconds so Excel can open. We are going to use the same theory but we will use SLEEP to wait until a precedent program/process is complete. The syntax is simple:

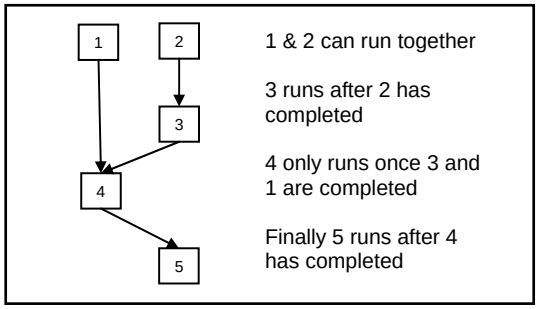
```
Data _null_;
   X=Sleep(10,1)
run;
```

The value in parenthesis is the number of seconds SAS will pause processing. In the above example SAS will pause for 10 seconds.

## STEP4 – PUTTING IT ALL TOGETHER

Now that the basics for the Shell Script, "Done" files and SLEEP command have been reviewed, it is time to put it all together. The example process will have 5 SAS programs and the run order and conditions will match the logic explained in the UNIX SHELL SCRIPT section.

**Figure 2 Program Order**




For this example we will need to add code to programs 1, 3, 4, and 5. In prog_1 delete any existing “Done” files and create a “Done” file once the program is completed

```
/* Code added the top of prog_1 */
Libname lib_done “/program_directory/”;
%macro delete_done(var1);
%if  %sysfunc(exist(lib_done.done_file&var1)) %then %do;
    proc datasets lib=lib_done;
           delete done_file&var1;
    run;
%end;
%mend delete_done;
%delete_done(prog_1);
run;

/* Code added to the bottom of prog_1 */
%macro create_done(var1);
Data lib_done.done_file&var1;
    done=”YES”;
    output;
run;
%mend create_done;
%create_done(prog_1);
run;
```

In prog_3 delete any existing “Done” files and create a “Done” file once the program is completed, the same as one did in prog_1.

```
/* Code added the top of prog_3 */
Libname lib_done “/program_directory/”;
%macro delete_done(var1);
%if  %sysfunc(exist(lib_done.done_file&var1)) %then %do;
    proc datasets lib=lib_done;
           delete done_file&var1;
    run;
%end;
%mend delete_done;
%delete_done(prog_3);
run;

/* Code added to the bottom of prog_3 */
%macro create_done(var1);
Data lib_done.done_file&var1;
    done=”YES”;
    output;
run;
%mend create_done;
%create_done(prog_3);
run;
```

Now we will edit prog_4 to look for the “Done” files from prog_1 and prog_3. To do this we need a SLEEP command

and a loop to check for the “Done” files. In this example one will look for the “Done” files every minute and will check for 2 hours. If after the 2 hours of looking for the “Done” files from prog_1 and prog_3 and not finding them prog_4 will not run. This time limit is put in so that prog_4 does not loop indefinitely. The time for the loop to look for the “Done” files should be long enough that prog_1 and prog_3 should have already completed, but note that you can have as long as you need for the time limit or not have one at all.

```
/*Code added to the top of prog_4 */
Libname lib_done “/program_directory/”;
%macro delete_done(var1);
%if  %sysfunc(exist(lib_done.done_file&var1)) %then %do;
    proc datasets lib=lib_done;
           delete done_file&var1;
    run;
%end;
%mend delete_done;
%delete_done(prog_4);
run;

/* Code to check for both Done files */
%global counter max;
%let counter=0;
%let max=120; /* minutes to check */
%macro done_check;
%if ((%sysfunc(exist(lib_done.done_prog_1)))=0 or
     (%sysfunc(exist(lib_done.done_prog_3)))=0) %then %do;

%do %until (((%sysfunc(exist(lib_done.done_prog_1)))=1 and
             (%sysfunc(exist(lib_done.done_prog_3)))=1)
            or (&counter+0 > &max+0));
*** Let SAS sleep for a minute ***;
data _null_;
    sleep_time=sleep(60,1);
run;
*** code to increment counter ***;
%let counter=%eval(&counter.+1);
%end;
%end;
*** Stop program if time out ***;
%if &counter+0 > &max+0 %then endsas;
%mend done_check;
%done_check;
run;
```

Macro, done_check, checks for the existence of prog_1 and prog_3 “Done” files. If both are not found then it will look every minute and recheck for the “Done” files. Once found the macro will stop and the rest of the code will run. If after the time limit the “Done” files are not found the last %if will cause the program to stop. When this happens no “Done” file will be created for prog_4. Remember one will also need to add code to create a “Done” file to prog_4 to tell prog_5 that prog_4 completed and did not “time out”.

```
/* Code added to the bottom of prog_4 */
%macro create_done(var1);
Data lib_done.done_file&var1;
    done=”YES”;
    output;
run;
%mend create_done;
%create_done(prog_4);
run;
```

In the start of prog_5, you need to check and see that prog_4 ran and did not “time out” looking for “Done” files. If prog_4 timed out there is no need to run prog_5. The code below checks to see if prog_4 ran and if not prog_5 will then terminate.

```
/* Code added to the top of prog_5 */
Libname lib_done “/program_directory/”;
%macro check(var1);
```

```
%if sysfunc(exist(lib_done.done_file&var1))=0 %then endsas;
%end;
%mend check;
%check(prog_4);
run;
```

The check macro looks for the existence of the prog_4 “Done” file. If not found the program will end. Now that all the additional code has been added to the programs one can run the shell script to run the entire process. This can be done at a UNIX prompt by running the batching.csh program (csh batching.csh)

## CONCLUSION

Projects often consist of multiple SAS programs, the dependences and order of processing can become increasingly complex. What we have discussed above is a simple solution that can be used anywhere SAS exists. Using just a UNIX shell script and a couple of commands within your programs, processes can be streamlined. The above was just a simple example of how to do this many more bells and whistles can be added to the UNIX script and the “Done” files to build out the process further to increase robustness and efficiency.

## APPENDIX

Below is the code one can use to create “Done” files that are text files

```
/* Code to create “Done” text file */
%macro create_done(var1);
filename donefile “/program_directory/done_&var1..txt”;
Data _NULL_;
    File donefile;
     Done=”YES”;
    put;
Run;
%mend create_done;
%create_done(program_name);
run;

/* Code to delete “Done” text file */
%macro delete_done(var1);
X “cd /program_directory/”;
X “rm –f done_&var1..txt”;
%mend delete_done;
%delete_done(program_name);
run;

/* Code to look for “Done” text files */
filename donfile1 “/program_directory/done_prog_1.txt”;
filename donfile2 “/program_directory/done_prog_2.txt”;
%global counter max;
%let counter=0;
%let max=120; /* minutes to check */
%macro done_check;
%if ((%sysfunc(fexist(donfile1)))=0 or
     (%sysfunc(fexist(donfile2)))=0) %then %do;

%do %until (((%sysfunc(fexist(donfile1)))=1 and
             (%sysfunc(fexist(donfile2)))=1)
            or (&counter+0 > &max+0));
*** Let SAS sleep for a minute ***;
data _null_;
    sleep_time=sleep(60,1);
run;
*** code to increment counter ***;
%let counter=%eval(&counter.+1);
%end;
%end;
*** Stop program if time out ***;
%if &counter+0 > &max+0 %then endsas;
%mend done_check;
%done_check;
run;
```

```
/* Code to stop program if “Done” text file is not found */
filename donefile “/program_directory/done_&var..txt”;
%macro check(var1);
%if sysfunc(fexist(&var1))=0 %then endsas;
%end;
%mend check;
%check(donefile);
run;
```

## REFERENCES

SAS Institute: SAS OnlineDoc, Version 9, 2005
SAS Institute: SAS Language Guide, Version 6.12

## CONTACT INFORMATION

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

Patrick Halpin
dunnhumby USA
302 West 3rd Street
Suite 300
Cincinnati, OH, 45202
Work Phone: 513-632-1170
E-mail: Patrick.halpin@us.dunnhumby.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.