User-Friendly Toolbox for Batch Processing within a UNIX® Interactive Display Manager Session

Jim Young, HCIA Inc., Ann Arbor, MI

Abstract

The interactive SAS® Display Manager and batch processing mode are the two main methods of running SAS programs, each of which has its advantages and disadvantages. We present a revisitation and expansion of a method of running the SAS system in a UNIX environment that combines many of the advantages of each. We created additional function keys and a customized toolbox. These features simplify adding the coding elements and issuing the commands that enable SAS programs to be edited interactively, yet submitted as quick running batch jobs. The point-and-click toolbox especially facilitates the ease of use and minimizes any learning curve. We find this tool invaluable in shortening the time required to develop and test new programs and assessing the impact of programming changes.

Introduction

Interactive display manager sessions allow easy access to data sets, program source code, and log and output information. The primary advantage of batch processing, on the other hand, is speed of execution. The 'interactive batch' technique is an approach that combines the best features of each method of running the SAS system. We can utilize display manager to examine data, source code and results, while processing in batch mode.

This technique was originally presented in John Blodgett's SUGI22 paper (#13) entitled "Combining Display Manager and Batch Modes Under UNIX[1]". A more complete discussion of the impetus behind this 'interactive batch' technique, as well as additional implementation considerations, can be found there. An electronic version of his paper can also be found at http://www.oseda.missouri.edu/mscdc/articles/dmbatch.txt.

An Interactive 'Batch-Ready' SAS Program

Let us begin by looking at the simple program "PRINT_ME.SAS" shown below. The 'interactive batch' method does not require any special tools (other than the SAS/FSP® module). The first 3 lines are the only thing unusual in the program -- they are also the keys to this method.

The X command allows us to send a command to the UNIX shell from within the SAS System. Using this, we change to the proper current directory. The second line assigns the program name to a macro variable so that it is now stored and accessible. This means that it is not necessary to retype the program name as long as we make use of the macro variable &pgm (it always resolves to the program name) which makes things quick and consistent. For example, line 3 becomes 'filename pgm print_me.sas' after resolution. Note that the macro variable &pgm is set not only for the rest of the program, but for the entire interactive session, or until it is changed. Additionally, we have a record of the program name and the path to its location hard coded into the program for future reference.

There are two optional external program calls at the end of the SAS program. Both the notify.sas and mailme.sas programs are just utility functions. The notify program serves to alert you (both visibly and audibly) when the program is complete, while the mailme program sends email to the UNIX user that invoked the program. We will come back to these later.

This paper furthers the 'interactive batch' method by illustrating a simple front end. Through the expanded assignment of function keys, and especially the addition of a toolbox designed specifically for this application, we made the execution of this technique more user-friendly. We did not make any substantive changes to the technique itself, only the tools used to write the programs.

<u>PRINT_ME.SAS:</u>

```
x cd ~/sascode;                        /*Issue a UNIX command to CD to proper directory*/
%let pgm=print_me;                     /*Store program name in macro variable*/
filename pgm "&pgm..sas";              /*Assign the fileref pgm to 'print_me.sas'*/

filename sascode '~/sascode';          /*Assign fileref to location of prepared SAS code*/
data name;                             /*From here down is a standard SAS program*/
 length fname $ 10 lname $ 10 logon $ 5;
 input fname lname logon phone;
 cards;
 Jim Young jyoun 7715
 ;
run;
proc print; run;

%include sascode(notify.sas);          /*Run the notify.sas program (which is stored in the ~/sascode directory)
                                       to alert us when print_me.sas completes.*/
%include sascode(mailme.sas);          /*Run the mailme.sas program (which is stored in the ~/sascode directory)
                                       to send email when print_me.sas completes.*/
```

Writing and Processing with Function Keys

Our first step towards creating a front end for the technique was to add more function keys to facilitate the inclusion of the most common programming elements used with this technique.

Programs written to take advantage of this 'interactive batch' technique generally begin with the three statements described above. Rather than typing them in each time, they can be included through the use of a function key (**Shift-F2**), as described in Table 1 below. This will place the lines

```
x cd PROJECT_PATH;
%let pgm=PGNAME;
filename pgm "&pgm..sas";
```

into your SAS Program Editor. (We chose the function keys out of convenience, but you can assign these commands to any open keys that appeal to you.)

The placeholders PROJECT_PATH (name of the desired project directory) and PGNAME (desired name of the SAS program) should be assigned, and then these commands need to be submitted. Submitting these commands will move us to the proper directory, assign a value to the macro variable, and define the fileref PGM. Immediately RECALL these lines before beginning to write the body of the program. For ease of use, submitting and recalling these initialization lines is assigned to a single function key (**Shift-F3**). Note that the function key issues the SUBTOP command as opposed to the SUB command. If you are editing an existing program you need to submit these three initialization lines, but you do not necessarily want to run the rest of the program. If you are writing a new program, then the SUBTOP command has the same effect as the SUB command. It is good practice to place any libname or filename statements immediately after the initialization lines. Note that **Shift-F3** will not process these additional lines, but they generally were not needed until we submitted the rest of the program. Of course, you can manually issue an additional SUBTOP <n> command if you need to define additional librefs or filerefs, where <n> is the requested number of lines. This can be useful to assign librefs for viewing resultant SAS data sets.

After submitting and recalling the initialization lines a new SAS program can be written, or an existing one altered. As you will soon see, the batch job runs the program from the UNIX prompt, not from within the display manager. Before submitting the program file, therefore, it must be saved. The macro variable that we have set up simplifies this process by remembering the program name. We only need to enter 'file pgm' (**Shift-F5**) because the fileref PGM was assigned when the initialization lines were submitted. Consequently, be sure to submit the initialization lines before you save your program or the SAS system will not know how to resolve the macro variable and assign the fileref. You will not likely get an error message, but you will probably end up saving your work to a file named 'pgm'. You can issue the FILENAME command to verify the value of the fileref before saving your program (more on this when we discuss the toolbox).

Now we need to submit the entire program (along with the first 3 recalled lines), but *not* to the interactive SAS session. We want to send this program to a new *batch* SAS session. Again, we'll make use of the X command that allows us to send SAS commands to the UNIX operating system. The SAS system will resolve the macro variable and submit the command to the UNIX shell (**Shift-F6**). The shell, in turn, will invoke the proper UNIX batch SAS program. We also chose to invoke this new SAS session as a background process. Since we never submitted the code from the display manager, the source code remains in the program editor window, ready to edit. Again, be sure to issue the 'file pgm' command to save any updates to the program immediately prior to submitting the 'interactive batch' program. Otherwise, you will run the same program again and again.

Because we submitted a batch job to the UNIX operating system the display manager knows nothing about the batch SAS job, but we would like to be told when it is done. This is where the optional 'notify.sas' program comes in. The SAS code to include the notify program can be placed into the current program using (**Shift-F4**). It issues a beep, as well as writes a message to both the LOG file and the Xterm window, when the program completes. You can also opt to have mail sent to you when the program completes (**Shift-F12**). We have found this mail feature to be especially useful for jobs that run for many hours because you can close your X-windows session and still be notified when the job is complete. The code for both the notify and mailback features can be found in the appendix at the end of this paper.

Table 1: Display Manager Function Key Assignments

Key	Definition	Purpose
Shift-F1	log off; listing hide; toolload sasuser.profile.dms; pgm;	Activate the Interactive Batch Toolbox
Shift-F2	inc '~/sascode/ibmode.sas'	Include the initialization lines
Shift-F3	subtop 3; recall;	Submit the initialization lines and recall source code
Shift-F4	inc '~/sascode/add_notify.sas'	Include the notification program
Shift-F5	file pgm	Save interactive program to UNIX file
Shift-F6	x sas &pgm &	Run saved file in background batch SAS session
Shift-F7	fslist "&pgm..log"	View the LOG file (LOG window) contents
Shift-F8	fslist "&pgm..lst" -cc	View the LST file (OUTPUT window) contents
Shift-F12	inc '~/sascode/add_mailme.sas'	Include the mailback program

The SAS program has been run, but since we did not submit an interactive job, the LOG and OUTPUT windows will show nothing. Yet we would like to see the results! The FSLIST command allows us to view non-SAS files in the SAS environment while retaining all the interactive elements (like the macro variable that we set). The 'interactive batch' method automatically creates the LOG(*.log) and OUTPUT(*.lst) files because it was a batch process. Resolution of the macro variable &pgm allows the SAS system to display the appropriate LOG (**Shift-F7**) and OUTPUT (**Shift-F8**) files, again without having to enter the program name. The underlying commands issued to the display manager are shown in Table 1. Since nothing gets written to the display manager LOG and OUTPUT windows when using this method, you may even want to hide these windows. Note that if you load the toolbox, as we will see below, the LOG and OUTPUT windows will be hidden for you.

Once we open an FLIST window to the LOG or OUTPUT files, the FSLIST function keys, described in Table 2 below, allow us to switch back and forth between the two files. For convenience, the function key assignments in FSLIST are the same as those in the display manager. The difference is that the display manager function keys open a window to either the LOG or OUTPUT file (depending on the function key) and reissuing the display manager command will open another FSLIST window. In contrast, the FSLIST function keys switch the contents of a given window back and forth between the LOG (**Shift-F7**) and OUTPUT (**Shift-F8**) files, but do not open any new windows. The advantage of this approach is that system resource use is minimized by maintaining only one window.

Table 2: FSLIST Function Key Assignments

Key	Definition	Purpose
Shift-F7	browse “&pgm..log”	View the LOG file (LOG window) contents
Shift-F8	browse “&pgm..lst” -cc	View the LST file (OUTPUT window) contents

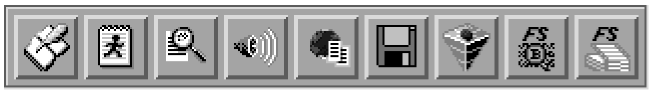
You can view the LOG and OUTPUT files (using the FSLIST commands) anytime while the batch job is running, but you should not save any changes to the source code during batch execution. After all, this is the program that the SAS system is running! These files can be viewed dynamically during execution, just as you can examine the LOG or OUTPUT files if you had invoked the background SAS session directly from the UNIX prompt. As you toggle back and forth between them you will get continually updated information in the window. The job can also be monitored or interrupted from the shell, just like any other UNIX job. You can also write another new, or edit an existing program while the job runs. Just be sure to resubmit (and check) the initialization lines so as not to overwrite any files.

Any errors can be fixed, the file resaved (because we are running the stored UNIX file, not what is on the screen), and the process repeated. We have access to display manager interactivity and editing tools, with batch speed processing and automatic LOG and OUTPUT file creation!

The Toolbox Implementation

To make this method truly efficient, however, we created a toolbox specifically designed to facilitate the interactive creation of the batch-processed SAS programs. To that end, the toolbox has an icon associated with each of the display manager function key definitions. The toolbox (shown in Figure 1 below) has icons to:

Figure 1: Interactive Batch Processing Toolbox



1. Include the initialization lines,
2. Submit and recall these lines,
3. Verify the fileref,
4. Add the notify feature to a program,
5. Add a mailback feature to the program,
6. Save the current program,
7. Submit the batch job to UNIX,
8. View the LOG file and
9. View the OUTPUT file.

The user invokes the SAS system in the standard display manager mode. The key combination (**Shift-F1**) switches you into ‘batch processing’ mode. In doing so, the LOG and OUTPUT windows will be hidden (since their function is replaced by viewing the external files using FSLIST) and the toolbox will be loaded. The function keys are all still available for users that might prefer this method, but the icons will now be available as well. Within the SAS system, bubble help pops up when the cursor is over any icon.

The icons eliminate the need to remember any of the display manager function key assignments. All of the steps described above can be quickly and easily performed in a point-and-click environment. The ‘interactive batch’ technique seems to be much less intimidating to new users because of the familiar point-and-click feel of the toolbox. The icons are arranged so that you generally travel left to right across the toolbox.

The first three steps are executed at the beginning of a session. At this stage the user includes the initialization lines, alters the placeholders and submits and recalls the initialization statements. Placed immediately after the submit and recall button, the toolbox also includes an icon designed to quickly double check the fileref -- it simply issues the FILENAME command. With this initial program prep work complete, the SAS program (or a section of the program) is then written.

The next four steps are then executed, though steps 4 and 5 are optional. The notify feature is very useful if repeated runs are expected, due to either error debugging or testing different programming options, so that you are notified when the program completes. As mentioned above, the mailback feature is utilized mainly for long running programs, but has also been very useful as a reminder that a program has completed after the display manager, or even the X-windows session, has been closed. Saving the program (by clicking the disc icon) is critical in order to run the most recent version of the program, and the submit button sends the UNIX shell command to invoke the actual background SAS session.

When the program completes, the user moves to steps 8 and 9. There are icons to view the LOG file and view the OUTPUT file using the FSLIST procedure. Once in FSLIST mode, the user can toggle back and forth between the LOS and OUTPUT files using the FSLIST function keys. Then, unless they are very good programmers, begins the iterative and interactive process of editing the program, saving the updated code, submitting a new batch job, and rechecking the LOG and OUTPUT files. All of these steps, except for the programming, can be quickly and easily completed with the click of an icon.

By default, the SAS toolbox changes as you move from display manager to FSP mode, and only one toolbox can be displayed at a time. This means that the ‘interactive batch’ toolbox disappears when you go to check your results. To change this default behavior, we set the following XResource: SAS.mergeToolBoxes:False. This can be accomplished for a particular session by invoking the SAS system with the -xrm option, or more permanently by adding this setting to either the config.sas612 or your SAS XResource file.

The specifics of creating a toolbox using the tool editor can be found in Chapter 15 “Unix Environments” of the *SAS Software: Changes and Enhancements, Release 6.11*[2] manual. An installable version of the toolbox for a UNIX machine running the SAS system in an X-windows environment can be obtained upon request. Because the availability of open function keys is user-dependent, however, we cannot safely supply a profile containing the pre-defined function keys without risking overwriting your current assignments.

Summary

We have found that the use of the interactive batch toolbox has greatly enhanced our productivity by allowing continual access to program source code, SAS data sets, and log and output files through a simple point-and-click interface that executes jobs as fast running batch processes. We have realized especially large productivity gains with iterative programs that regularly require writing code, running the program, examining the results, and altering the code accordingly. Most programs fall into this category during development, for at least some period of time. Display manager users have adopted this technique more quickly than those that execute the SAS system in batch mode. But though the benefits are more readily apparent from within a display manager session, the attractiveness of being able to browse SAS data sets, and examine results with the click of a button (rather than calling them up in a UNIX editor) has converted batch-processing SAS system users as well.

Appendix: SAS Code for Utility Programs

NOTIFY.SAS

```
options nonotes nosource no$syntaxcheck obs=max;
data _null_;
 call sound(5,5);
 file log;
  put ' ';
  put "DONE: &pgm..sas was run on &sysday &sysdate at
&systime with V&sysver..";
file stderr;
  put "DONE: &pgm..sas was run on &sysday &sysdate at
&systime with V&sysver..";
run;
options notes source;
```

MAILME.SAS

```
*Get login name of user;
%let mailto=%sysget(USER);

*Send simple message to specified user;
filename mailme email "&mailto.@company.com"
subject='Notification of completed SAS job';
data _null_;
 file mailme;
    put "The SAS program %upcase(&pgm) has completed!";
    put "It was run on &sysday &sysdate at &systime with
V&sysver.." ;
run;
```

[1]Blodgett John. (1997), “Combining Display Manager and Batch Mode Under UNIX,” *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, 22, 64-67.

[2]SAS Institute Inc. (1995), *SAS Software: Changes and Enhancements, Release 6.11*, Cary, NC: SAS Institute Inc.

Jim Young
HCIA Inc.
24 Frank Lloyd Wright Drive
P.O. Box 303
Ann Arbor, MI 48106-0303
Phone: (734) 669-7715
FAX: (734) 996-8859
E-mail: jyoun@hcia.com