Applications for Running DOS Commands within SAS

Na Li, Pharmacyclics, Sunnyvale, CA

ABSTRACT

Using the X statement in SAS® to issue DOS commands and initiate other Windows applications can be very useful and efficient in managing, comparing and scheduling SAS programs and setting data set and program permission attributes. This paper details some application examples for programmatically copying programs from one directory to another, creating directories based on the contents of the data, scheduling programs to execute in sequence, comparing programs between directories, and setting up read/write permissions. The advantages of using the X statement and some usage considerations are also discussed.

INTRODUCTION

Within SAS the X statement can be submitted to execute operating system commands. In the Windows operating system, using the X statement allows SAS to perform DOS commands and initiate Windows applications such as Word and Excel from within a SAS program. Using the X statement not only provides the convenience of programmatically and automatically managing files but also provides a tracking document to trace the file source and track the process, which can be a difficult task when the process is done manually in the manner of click and drag. Executing a DOS command from within SAS is relatively simple. However, this paper reveals some efficiency techniques and useful tips that are not commonly documented. The examples in this paper are processed by using the PC MS Windows XP operating system and the SAS version 8.2 is used to execute the programs.

COPY FILES

The syntax for issuing a DOS Command from within SAS is simply, X 'DOS command'.
Here is an example for copying an Excel file with the name of PGMTEST from C:\My Documents to F:\STUDY101.

```
X  'copy  "C:\My Documents\PGMTEST.XLS" "F:\STUDY101\PGMTEST.XLS" ';
```

Where C: and F: are mapped drive letters, UNC (Universal Naming Convention) paths may be substituted for mapped drive letters.

Although basic copying command is X 'copy A B'; , when the file path has embedded blank space, the quotes need to be used to allow the DOS command to work appropriately. The above codes should be X 'copy "C:\study101 output\A.sas" B'; . For the files that have macro variable(s) and contain embedded blank spaces in the file path, the copy statement need to be like: X "%str(copy %"file location\&outname..xls %" %"file new location\&outname..xls %")"; . The double quotation marks need to be used rather than the single quotation marks.

For example, a routine monthly task copying two updated Excel files to a central location, along with other files and SAS data set, the following SAS codes can be used. Only the macro variable needs to be modified each month.

```
%LET mth=transfer December 2004;

%macro doit;
x "%str(copy %"N:\data transfer\&mth\Data\ptqcs.xls%"
            %"N:\data transfer\central\ptqcs.xls%" )";
x "%str(copy %"N:\data transfer\&mth\Data\Report.xls%"
            %"N:\data transfer\central\Report.xls%" )";
%mend;

%doit;
```

A SAS program to execute the X statement to copy files can be very useful for performing a routine task requiring DOS related commands. To do so can avoid manual click and drag errors and save time. The program (unchanged or archived) also can serve as a document to track the source of the original file(s).

SET UP PERMISSION ATTRIBUTES

The X statement can also be used to set up read/write/execute permissions. To set the file as read-only, the syntax is like: X attrib +r "C:\filename.extension". In order to set the permissions to multiple files, the syntax is:

```
%macro mkRonly;
  %do i=1 %to &nfile;
       *** nfile is a macro variable indicate the total counts of the number of data set;
    x attrib +r "file location\&&ds&i..sd2";
         ***&&ds&i is also a macro variable, containing the names of the SAS data set;
  %end;
%mend mkRonly;
```

Using the above codes can set the attributes of serial files in one program so that time can be saved in comparison to clicking on the property of each file and checking the Read-only Box (see below, the code is equivalent to checking the Read-only box in the Attributes section). Additionally, the above method can avoid errors of missing some files.
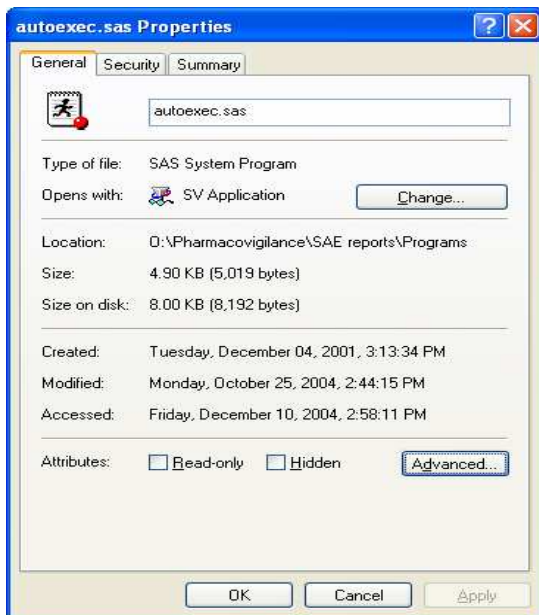
To remove the Read-only restriction (X attrib -r "C:\filename.extension") to serial files, the following program can be used:

```
%macro rmRonly;
  %do i=1 %to &nfile;
      x attrib -r "file location\&&ds&i..sd2";
  %end;
%mend rmRonly;
```

CREATE DIRECTORIES BASED ON THE DATA

The **X** statement can also be used to create directories based on the data. The following code creates subdirectories for each patient in the PGM1 data set under the designated file location on the C drive.

```
Data fordir; set PGM1 end=eof; by ptid;
  Retain seq 0;
  If first.ptid then do;
     Seq=seq+1;
     Call symput('pt'||compress(put(seq, best8.)), trim(left(put(ptid, 3.))) );
         *** assume patient ID (ptid) has 3 digits;
  End;
  If eof then do;
     Call symput('totpts', compress(put(_n_, best8.)) );
  End;
Run;
```

```
%macro runit;
  %do i=1 %to &totpts;
     X "%str(md %"c:\file location\&&pt&i%" )";
  %end;
 %mend runit;
```

First of all, the Patient IDs are retrieved as macro variables. Then by using the X command MD (make directory) in the macro do loop, the directories corresponding to the Patient ID are created.

Since the X commands are global SAS statements, they cannot be executed conditionally within a Data Step. In order to make directories in a conditional manner, the CALL SYSTEM command needs to be used. To create subdirectories for each patient ID under the C:\temp directory when the condition of the flag equals 'y', the syntax should be:

```
data _null_;
  set fordir;

  if flag='y' then
   do;
    commline='md c:\temp\'||ptid;
    call system (commline);  *** call system command;
   end;
 run;
```

However, X commands can be conditionally executed when they are a part of %if %end block within a macro program.

RETRIEVE ALL THE FILE INFORMATION FROM A DIRECTORY AND SCHEDULE PROGRAMS TO RUN

To schedule programs to be executed in a sequential order, we need to create a text file that contains file names and creation dates of all the files in a directory, including any subdirectories. For example, using the syntax of X 'dir/s C:\TEMP > C:\OUT\prglist.txt ' creates a directory listing of all the programs under the TEMP folder (use the /s to include all the subdirectories) and writes it to a text file called PRGLIST under the OUT directory in the C drive.

To be able to schedule the job to run in a sequential order, a .BAT file needs to be created based on the text file (e.g. PRGLIST). The program is like:

```
Filename pgmlist "c:\out\prglist.txt";

Data runit;
   Infile pgmlist length=len;
   Input line $varying $175. len;

  Startfrm=index(line, '.sas');

  If startfrm >0 then do;
   Pgm=scan(substr(line,1,startfrm), -1);
   Output;
  End;
 Run;

Data runit;
   Set runit;
     Loc="start/w C:\sas\sasv8\sas.exe -sysin c:\out";
        *** the location of the sas.exe is different based on the folder structure;
     Fullpath=loc||compress(pgm)||'.sas';
Run;

Filename runit "c:\out\run_pgm.bat";
Data _null_;
   File runit;
     Set runit;
     Put @1 fullpath;
```

```
Run;
```

FILE COMPARISON

Comparing among programs can be achieved literally by using X fc a.sas b.sas>>comp.txt. The DOS command FC is used to do the file comparison. When using >comp.txt, it will save the content to a newly created file (comp.txt). Using >> instead of > will concatenate the rest of the file comparison to the same file comp.txt. Some DOS commands may not work appropriately when they are used as the X statement, such as FC and FIND commands. Putting COMMAND/C (start a new DOS command processor) before the DOS command usually can resolve the problem. The syntax of performing this kind of command within SAS is:

```
X "command/c fc c:\temp\test1.sas c:\temp\test2.sas > c:\temp\comp.txt";
```

CONTROLLING HOW THE X STATEMENT WORKS

By default, when using the X statement to execute DOS commands, the SAS session is frozen until EXIT is entered at the DOS prompt or the Windows application is closed to reactivate the SAS session. To be able to automatically close the DOS window after executing all the DOS commands and continue the SAS session, the NOXWAIT option needs to be stated before the X statement. When NOXWAIT is called, the DOS window will open, execute DOS commands and close, appearing like a flash. On the other hand, when viewing the results of the DOS command is preferred, keep the option as the default setting. In order to keep the SAS session active, while the DOS session or Windows application is still active, set the option to NOXSYNC. If waiting for the completion of the DOS commands is necessary, the following code should be added to allow the DOS commands to be fully executed. In this particular example, the codes let the SAS session sleep for 5 seconds (can be changed to any suitable seconds) to allow the DOS commands finish the DOS execution before the next step execution in the SAS program. Usually prior to the initiation of the Windows applications by using the X statement, the code below is added.

```
 Options noxwait noxsync;

***the X statement is used to invoke the EXCEL application to run;
***initiate EXCEL, EXCEL HAS TO BE CLOSED ********;
***location of the Excel.exe varies and depends on the folder structure;

x '"C:\Program Files\Microsoft Office\Office10\EXCEL.EXE"';

 Data _null_;
    Wait_sec=sleep(5);
 Run;
```

CONCLUSION

Using the X command is the only way to get certain things done in a SAS program. For example, SAS code using Proc Datasets and Proc Copy can delete data sets and copy data sets or catalogs, but they cannot manipulate non-SAS related files. Only the X command can do tasks such as set file attributes, rename, move, or copy non-SAS related files within a SAS program.

Besides using the X statement to run the X commands in the SAS session, there are other methods, such as CALL SYSTEM routine, %SYSEXEC statement, and SYSTASK statement. The CALL SYSTEM routine has to be executed within Data Step, while the X statement can be executed either outside or within Data Step. Both the CALL SYSTEM routine and the X statement, by default, are executed synchronously, which means the execution of the DOS commands or Windows applications needs to be conducted before the continuation execution in SAS session. But SYSTASK is executed asynchronously and has more options.

The syntax and usage limitation of the above methods may vary depending on the PC operating system (i.e. the syntax might be slightly different from WS NT, WS 98, and WS XP) and the version of the SAS. For example, the usage of the X statement before SAS version 9 has the length limitation (462 characters). When the statement is too long, it cannot be executed. To avoid the problem, put the X command in a .bat file instead.

REFERENCES

Chen, Ling (2003), "Run All Your SAS® Programs in One Program: Automatically" Proceedings of the NorthEast SAS Users Group Conference, Paper PO010.

SAS Institute Inc. (2000), "Running DOS or Windows Commands from within SAS" SAS OnlineDoc, Version 8, Cary, NC: SAS Institute, Inc.

ACKNOWLEDGMENTS

CONTACT INFORMATION

If you have any comments and questions regarding this paper, please contact:

Na Li
Pharmacyclics, Inc.
995 East Arques Avenue
Sunnyvale, CA 94089
(408) 990-7293 (work)
E-mail: nli@pcyc.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.