Some Things Mainframe SAS Programmers Need to Know to Use SAS On the PC

Clarence Wm. Jackson, CSQA
City of Dallas

Abstract

This paper is intended to assist SAS mainframe programmers in making the transition to writing SAS programs for the PC and client server environment. Since the core SAS language is the same regardless of the operating environment in which it runs, and only differs in the operating system's host interface, special attention will be given to how to access the PC (Windows) host files for input and directing output, and relating it to MVS JCL.

Introduction and Background

Since most people with "negative SAS dates" started using SAS on mainframes in batch or interactive mode, and are now faced with "PC data" and need to be able to access the data in the PC environment, a basic understanding of the PC world is required (unless you FTP your data to the mainframe, which will not be discussed in detail).

While most mainframe programmers became "batch" programmers, those with interactive experience will find that SAS on the PC is the same basic windowing environment of the mainframe, hyped and powered by the GUI of the Windows host. However, the mainframe host is different from the 'new' host.

Understanding how to access data on the mainframe requires an understanding of job control language (JCL), CLIST, or REXX to allocate and access mainframe files of various organizational types (VSAM, QSAM, ISAM. BDAM, PDS, tape, DASD, etc) to read the record by their format (FB, FBV, VB, etc), and define the data elements that are in various formats (fixed vs variable length, packed, zoned, etc), and also interact with other mainframe host information. Since all files are known to the mainframe, the logical path is the file's name.

Mainframe data files are accessed thru batch JCL or interactive allocations with the file information supplied through "DD" options for the files, or since V6 of SAS, using "LIBNAME/FILENAME" options for dynamic allocations, which is the same for Windows, except for the file naming and formats.

The same level of understanding for the Windows Host is required to access data in the PC client server host environment, but where are the files, and how are they accessed? What format is the data in, and how is it defined? The data could be on the same workstation, on a server within the LAN/WAN, or on a floppy or zip disk. Does it matter? Understanding file organizations in this environment goes a long way toward making the transition from the mainframe.

One of the central things to understand about the Windows environment is in knowing how files are structured and organized in folders/directories, versus how files are organized on the mainframes.

In the Windows environment, files have a name and an extension that indicates the type of file. Usually the extension is registered to Windows to be opened by a specific application, such as '*.SAS' files being opened by the SAS application. The "logical path" is the location of the file, such that the storage drive, directory, and file's name and extension will provide direct access to the file. So, to locate a file, you will need to know this info. The format is usually *'drive:/directory/subdirectory/file.extension'.*

SAS Program Example

The following SAS program may be familiar to lots of early SAS users, as it was used in the SAS Beginning Programming Guide, and I have used it to start ALC and COBOL programmers new to using SAS.

```
* SAMPLE SAS PROGRAM;
```

C

```
DATA CLASS;
  INFILE CARDS;
INPUT NAME $ 1-10
      SEX  $ 12
      AGE    14-15
      HEIGHT 17-18
      WEIGHT 20-22;
CARDS;
ALFRED     M 14 69 112
ALICE      F 13 56 84
BERNADETT  F 13 65 98
BARBARA    F 14 63 102
HENRY      M 14 63 102
JAMES      M 12 57 83
JANE       F 12 59 85
JANET      F 15 62 112
JEFFREY    M 13 62 84
JOHN       M 12 59 99
JOYCE      F 11 51 50
JUDY       F 14 64 90
LOUISE     F 12 56 77
MARY       F 15 66 112
PHILLIP    M 16 72 150
ROBERT     M 12 64 128
RONALD     M 15 67 133
THOMAS     M 11 57 85
WILLIAM    M 15 66 112
```

A

D

```
;
PROC PRINT;
RUN;
```

This program will run in any environment, with only the means of executing within the host environment changing. For a mainframe programmer, to run this job in batch would require job control language (JCL) that would look something like this:


```
//SAMPLE  JOB (TEST,XXXX),CJAC.4DS,CLASS=S,NOTIFY=&SYSUID
//STEP1   EXEC SAS
//SYSIN   DD *
{ SAS PROGRAM Statements}
//
```

If the program is stored in a PDS or other file, the JCL would need to point to the file containing the SAS program, and would look like this:


```
//SAMPLE  JOB (TEST,XXXX),CJAC.4DS,CLASS=S,NOTIFY=&SYSUID
//STEP1   EXEC SAS
//SYSIN   DD DSN=PDSFILE.WITH.CODE(sample),DISP=SHR
//
```

For a Windows programmer, the SAS program would be stored in a file with the '*.SAS' extension, and to execute it on a machine with SAS would require only to double click the file. Another method is to create a batch file (*.BAT) with the following line command, or typed at a DOS prompt:

```
C:\SAS_Folder\SAS.EXE C:CJACKSON\SASSTUFF\CLASSWRK\SAMPLE.SAS
```

The Windows programmer could also start SAS, open the file in the program manager, and click 'RUN-SUBMIT' the job. As you can tell, there are many ways to run the program in the Windows environment.

Let's add an output file to our program to store the data, and add a line to the program to write the file just before the 'CARDS' statement. The line to write the file would look like this:


```
FILE classout;
PUT @1 _INFILE_;
```

The file 'class' would need to be defined. Newer SAS programmers would include the "FILENAME" statement inside the SAS program, while the natural tendency of earlier mainframe SAS programmers would be to add a "DD" reference in the JCL. SAS allows the "FILENAME" statement in all current versions of SAS, and its use should be encouraged for portability of SAS programs. Using this format also allows for dynamic allocation of files, a plus when using the SAS Macro language.

The notations for adding the 'class' file reference are as follows:

MVS JCL	//CLASS DD DISP=SHR,DSN=DATAFILE.CLASS or //CLASS DD DISP=(,PASS),DSN=PDS.DATAFILE(CLASSOUT)	B
Window	FILENAME CLASS "C:CJACKSON\SASSTUFF\ CLASS.TXT";	C
MVS	FILENAME CLASS "PDS.DATAFILE(CLASS)" DISP=NEW;	

The insertion point for the above code modification is notated. Because using the 'FILENAME' statement (C) allows both versions of the code to reference the files internal to the SAS program. Using this format and "LIBNAME" statements instead of JCL DD statements will help make your programs more portable, and your programming more standard between the mainframe and Windows environments.

Where are the data, format, and macro libraries?

There are numerous other file and library types in SAS, but for the purpose of this section, I will cover SAS format, and macro libraries and their general organization on the mainframe, and how they are organized in the Windows environment.

Under OS mainframe format and macro libraries are a type of partitioned data set (PDS), and are arranged with each format and macro stored as separate members of the PDS file. This means that SAS uses the base file name, and references the members as needed. For instance, the PDS file may be named "PDS.SASLIB", and contain macros as members A, B, C, and D. In your SAS program, you would use the SAS "LIBNAME" statement to reference "PDS.SASLIB", and SAS would read A, B, C, or D as needed.

In the Windows environment, the libraries would be directories or folders, and would contain separate files for each format and macro. To reference these, you would refer SAS to the folder or directory name. So for instance, the folder "c:\mysaslib" would contain macros as files A.sas, B.sas, C.sas, and D.sas. In your SAS program, you would use the SAS "LIBNAME" statement to reference "c:\mysaslib", and SAS would read A, B, C, or D as needed.

Window	LIBNAME MyMacros "c:\mysaslib";
MVS	LIBNAME MyMacros "PDS.SASLIB" DISP=SHR;

Another way to define your libraries in the Windows side is to use the Library Dialog box. There are other tools that make management of your SAS libraries much easier on the Windows and mainframe interactive side that aren't possible using JCL.

GUI Tools

Some of the neat tools that are available on the Windows side that are really nice to use is the import wizard, export wizard, and the many windows such as the Program Editor, Log, Output, and Options windows. These make it very easy to run SAS in the

Windows environment. Also available are tools to allow you to read SAS data sets, see available libraries and files, and other fun tools that allow for productive SAS sessions.

One of my favorites is the import wizard. This tool will read standard file formats such as *.txt, *.csv, and other delimited file, create a SAS data step to read the file, and run the created SAS code. The neat thing about this is that you can 'RECALL' the job that was submitted, modify to your specification, and save the file as a SAS program for later use.

Conclusion

The move from mainframe to client server/distributed environments for enterprise data applications is one of the reasons for needing to use SAS in a Windows environment. The core of the SAS language is the same on all operating system, but differs only in how files are handled and data stored. Newer users of SAS have come to the software from the Windows side, while long time users started on the mainframe.

This paper was intended to give an overview of the differences in submitting SAS code in mainframe OS and Windows, and to assist in bridging the understanding of where and how to do basic SAS submissions in each. Complete tutorials for Windows have been written and are shipped with the software to the site representatives in the book *"Painless Windows"* by Jodie Gilmore, a very good reference that should be used along with the SAS Companion Guides.

Author Contact info:
Clarence Wm. Jackson, CSQA
City of Dallas, Communication & Information Services
Manager of Change Management, Quality Assurance, and IT Disaster Recovery
1500 Marilla 4DS
Dallas, TX 75201
cljacks@ci.dallas.tx.us (work)
CJac@compuserve.com (home)