PharmaSUG 2013 - Paper CC21

Dynamic Project Setup and Programming Using SAS Automatic Macro Variables and Environment Variables

Gary E. Moore, Moore Computing Services, Inc., Little Rock, Arkansas

ABSTRACT

Often, we find ourselves working on multiple projects with multiple programmers. The projects are located in various directories. Some programmers prefer a particular editor, while others prefer working in the Display Manager System. All of the work, no matter how it was developed, will need to come together at completion and be updated. In this working environment it is important to have a flexible way to assign library names and create programs that will work no matter where the project is located or which programmer is working on it.

This paper presents dynamic code using SAS automatic macro variables and environment variables that help resolve the conflicts that can occur in this working environment.

INTRODUCTION

There are common aspects of every working environment; things that need to be defined and/or known for every project. Among these are: The directories that contain the programs, macros, output (reporting, logs, listings), and the data all need to be defined. Is the program being run interactive or non-interactive? Where and when is the SAS program being run?

Using automatic macro variables and environment variables can help to achieve these goals. Automatic macro variables are created by the macro processor at SAS invocation and supply information related to the SAS session. Environment variables are preset in the operating system or by SAS at invocation and supply information about the operating environment. Using SAS Automatic Macro Variables and Environment variables an initialization program can be created that dynamically determines where and how a SAS program is ran.

INTERACTIVE OR NON-INTERACTIVE

Is the program being run interactively or non-interactively? This can be determined by using the SAS automatic variable SYSPROCESSNAME. If this is interactive, the value will be **DMS Process #**. If this is non-interactive, the value will be **Program "Program Path and Name"**. The following code creates a macro variable, proctype, that will contain the value DMS for a SAS interactive session and Program for a non-interactive session.

```
%let proctype = %scan(&SYSPROCESSNAME,1,%str( ));
```

PROGRAM PATH AND FILENAME

What is the path and filename of the SAS program? SYSPROCESSNAME is not the only variable that is assigned different values if the session is interactive or non-interactive. Non-interactive, the SYSIN system option has the program path and filename value, but not interactive. Alternatively, the environment variable, SAS_EXECFILEPATH, has the path and filename value interactively, but non-interactively.

```
/* Define macro variable for the path and filename */
%if &proctype = DMS
     %then %let pathpgmref=%sysget(SAS_EXECFILEPATH);
     %else %let pathpgmref=%sysfunc(GetOption(SYSIN));

/* Define separate macro variables for the filename.extension, */
/* filename, and extension                                     */

/* Reverse path and filename to scan from the front (back)     */
%let pathpgmref = %sysfunc(reverse(&pathpgmref));

%let pgmextref = %scan(&pathpgmref,1,\);
%let extref = %scan(&pgmextref,1,.);

/* Reverse back to proper order */
%let pathpgmref = %sysfunc(reverse(&pathpgmref));
%let pgmextref = %sysfunc(reverse(&pgmextref));
```

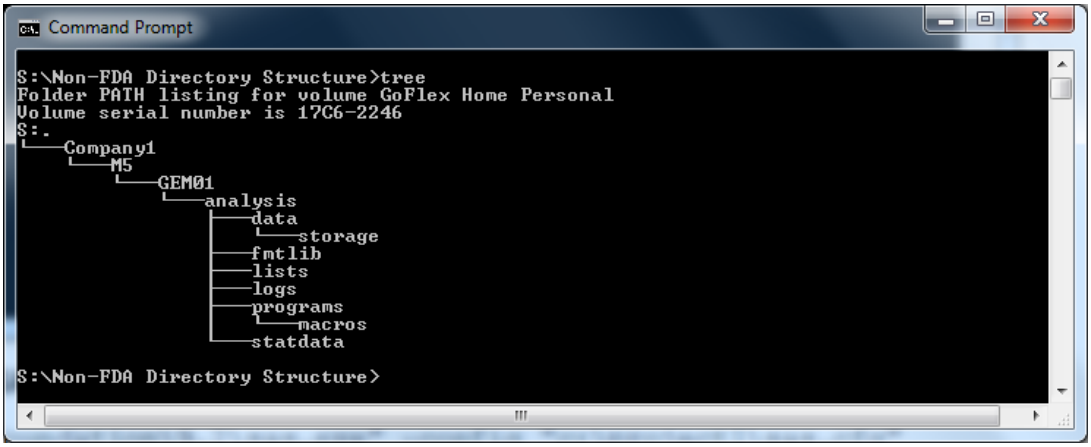
```
%let extref = %sysfunc(reverse(&extref));
%let pgmref = %scan(&pgmextref,1,.);

/* Remove filename to create program directory macro variable */
%let temp = %eval(%index(&pathpgmref,&pgmref) - 1);
%let pathref = %substr(&pathpgmref,1,&temp);

%put &pathpgmref;
%put &pathref;
%put &pgmextref;
%put &pgmref;
%put &extref;
```

An example of the %put to the log file will contain information similar to:
S:\Company1\MG\GEM01\analysis\programs\envtest.sas; S:\Company1\MG\GEM01\analysis\programs\; envtest.sas; envtest; sas.

DIRECTORIES, LIBNAMES, AND FILENAMES

Since most companies maintain a similar directory structure for their clinical trial reporting, the libname definitions of common directories should be independent of which project you are working on. Since finding the path of the program file, all of the other directories should be located around the program file directory.

If the data and format directories are located on adjacent levels as the program directory, then find the path of the directory one directory up and define their libnames. If the SAS macro library is one level down, then define the filename for the macro library using the program directory.



Output 1. Directory Tree Structure

```
/* Reverse path and filename to scan from the front (back) */
%let pathref = %sysfunc(reverse(&pathref));
%let uppathref = %scan(&pathref,1,\);

/* Reverse back to proper order */
%let pathref = %sysfunc(reverse(&pathref));
%let uppathref = %sysfunc(reverse(&uppathref));

/* Remove current directory to create path to parent directory macro variable
*/
%let temp = %eval(%index(&pathref,&uppathref) - 1);
%let uppathref = %substr(&pathref,1,&temp);

%put &uppathref;

/* Define libnames for data */
libname rawdata "&uppathref\data" access=readonly;
libname statdata "&uppathref\statdata";
```

```
libname storage "&uppathref\data\storage";

/* Define libnames for formats and search order */
libname fmtlib "&uppathref\fmtlib";
options fmtsearch=(work indata);

/* Define the filename for macro library and search order */
filename statmacs "&pathref\macros";
options sasautos=statmacs;
```

DATE AND TIME OF EXECUTION

When did all of this happen? Using SYSFUNC, allows access to the SAS date and time functions, assign them to a macro variable, and format them all at the same time.

```
%let execdttm = %SYSFUNC(date(), YYMMDD10.) %SYSFUNC(time(), time5.);
```

PROGRAMMING

The previous example has been the setup or initialization of the project, but what about the development of program code? It would be nice if everyone conformed to the same method of development, but that is seldom the case. Even while adhering to programming SOPs, personal preferences can be involved in the development of program code. Developing code in the Enhanced Editor and running interactively in the DMS can create program code that does not work in a non-interactive environment. Code that was developed in a non-interactive environment may not have the same desired effect when ran interactively in the DMS.

To achieve code that works in both, develop the program with the awareness of whether it is being ran interactively or non-interactively.

```
%if &proctype = DMS %then
     %do;
     filename LOGFILE  "&uppathref\logs\&pgmref..LOG";
     filename OUTFILE  "&uppathref\lists\&pgmref..LST";
     dm 'log; file LOGFILE replace';
     dm 'out; file OUTFILE replace';
     %end;
```

CONCLUSION

Using SAS automatic macro variables and environment variables you can create dynamic and independent setup and program code that can be used on multiple projects with similar structure.

REFERENCES

SAS 9.2 Macro Language Reference

SAS 9.2 Companion for Windows

CONTACT INFORMATION

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

Name:	Gary E. Moore
Enterprise:	Moore Computing Services, Inc.
Address:	2006 Beckenham Cove
City, State ZIP:	Little Rock, AR 72212
Work Phone:	501-225-8689
E-mail:	gary@moorecsi.net
Web:	www.moorecsi.net