Good Programming Practice when Working Across PC SAS and UNIX SAS

James Zhao – Merck & Co., Inc.

Abstract

This paper examines good programming practices to address common issues across PC SAS and UNIX SAS. They include: 1, Define forward or back slash for file paths in libname, filename, and %include statements consistently or dynamically; 2, Avoid the use of mixed case for file path and dataset names; 3, Avoid the use of X commands in SAS code to execute statements on the operating system which only works on Windows but not on UNIX; 4. Use appropriate SAS rounding function for numeric variables to avoid different results when dealing with 64 bit operating systems and 32 bit systems; 5, Use SAS Enterprise Guide to access and run PC SAS programs in UNIX efficiently.

Introduction

Many SAS users are working across multiple platforms, commonly combining Windows and UNIX environments. Often times SAS code developed on one platform (e.g. PC) might not work on another platform (e.g. UNIX). Portability is not only working across multi-platform environments, but also about making programs easier to be used across projects, companies, or clients and vendors. Many SAS programmers use PC SAS as their default working environment due to it convenience and simplicity. However, they are concerned with the use of UNIX SAS and how to run programs created on the PC in UNIX or vise verse. They want to create SAS programs that would run on either platform without modifications. This goal can be achieved through multiple approaches including the use of good programming practices, the use of SAS Enterprise Guide, and so on.

Good Programming Practice 1: Define forward or back slash for file path consistently or dynamically

Generally PC SAS can handle both forward and back slash while UNIX can only handle forward slash. Consequently, we need to avoid defining file paths in libname, filename, and %include statements using platform specific syntax such as forward slash (in UNIX) or back slash (in PC SAS). A good programming practice is to either always use forward slash across the entire SAS program consistently which will work on both PC SAS and UNIX, or to define the slash dynamically by using SAS automatic macro variable &SYSSCP or &SYSSCPL.

&SYSSCP and &SYSSCPL resolve to an abbreviation of the name of the operating environment. In some cases, &SYSSCPL provides a more specific value than &SYSSCP. You could use &SYSSCP and &SYSSCPL to check the operating environment to execute appropriate system commands. For example, we can run the below macro at the beginning of a SAS program to solve the problem caused by slash:

```
%macro slash;

%global slash;

%*** If the program is run on Windows;

%if (&sysscp = WIN) %then %do;
%let slash = %str(\);
%end;

%*** If the program is run on UNIX;

%else %do;
%let slash = %str(/);
%end;
%mend;

%slash;

*** define libname using &slash;

libname indata "&protpath.&slash.dataanalysis";
```

Good Programming Practice 2: Avoid the use of mixed case for file path and dataset names

PC SAS is not case sensitive. We are free to use either upper or lower case for file path or dataset names. However, UNIX is case sensitive. As a result, when there is a potential to run a SAS program on either PC SAS or UNIX, it is imperative not to use mixed upper case or lower case for file path or dataset names. A good practice is to try to only use lower case across the entire program. However, using mixed case for variable names is fine in both platforms.

Good Programming Practice 3: Avoid the use of X commands in SAS code

One of the powerful features of PC SAS is the X commends a user can insert and execute within SAS code to perform operating system functions such as deleting a file, renaming a file, and so on. However, this feature is operating system dependent. It only works on Windows but not on UNIX.

The below sample code in PC SAS can delete a file called delete_me.doc:

```
x "del delete_me.doc";
```

In Contrast, on UNIX, we can use the CALL SYSTEM routine inside a DATA step to execute operating system command. For example, the below code will change the current directory of the SAS session:

```
data _null_;
call system ('cd /users/zhaoyi/pgm');
run;
```

Good Programming Practice 4: Use appropriate SAS rounding function for numeric variables

When there is a potential to run a SAS program on different PCs with 64 bit operating systems and 32 bit systems, a careful design of using rounding function for numeric variables is needed to avoid different results which are difficult to detect. Below is an example of using function round(). If round() is not used, the result of EQUAL could be different when the program is run on a 32 bit system or on a 64 bit system since the numeric results can be twice as precise in 64 bit than in 32 bit system.

```
data two;
set one;
a = round(dose1/weight, 0.01);
b = round(dose2/weight, 0.01);
if a = b then equal = ‘Y’;
run;
```

Good Programming Practice 5: Use SAS Enterprise Guide to access and run PC SAS programs on UNIX effectively

SAS Enterprise Guide offers an array of benefits thanks to its GUI development environment. It makes all cross-platform work much easier because Enterprise Guide does not care where the SAS code resides and runs. You can write a program, save the .sas file to a local PC or to a remote UNIX server, run the program on the remote server, review the log, output, and debug, all of which are done through an Enterprise Guide session running on PC. More impressively, even if you choose to store .sas files only on your PC, you can still use Enterprise Guide to execute the code on a remote server.

Using Enterprise Guide to create a SAS program and save it to UNIX server is simple and straight-forward:

1. Open Enterprise Guide
2. File -> New Program
3. Develop program
4. Save As and indicate it will be saved to the UNIX server

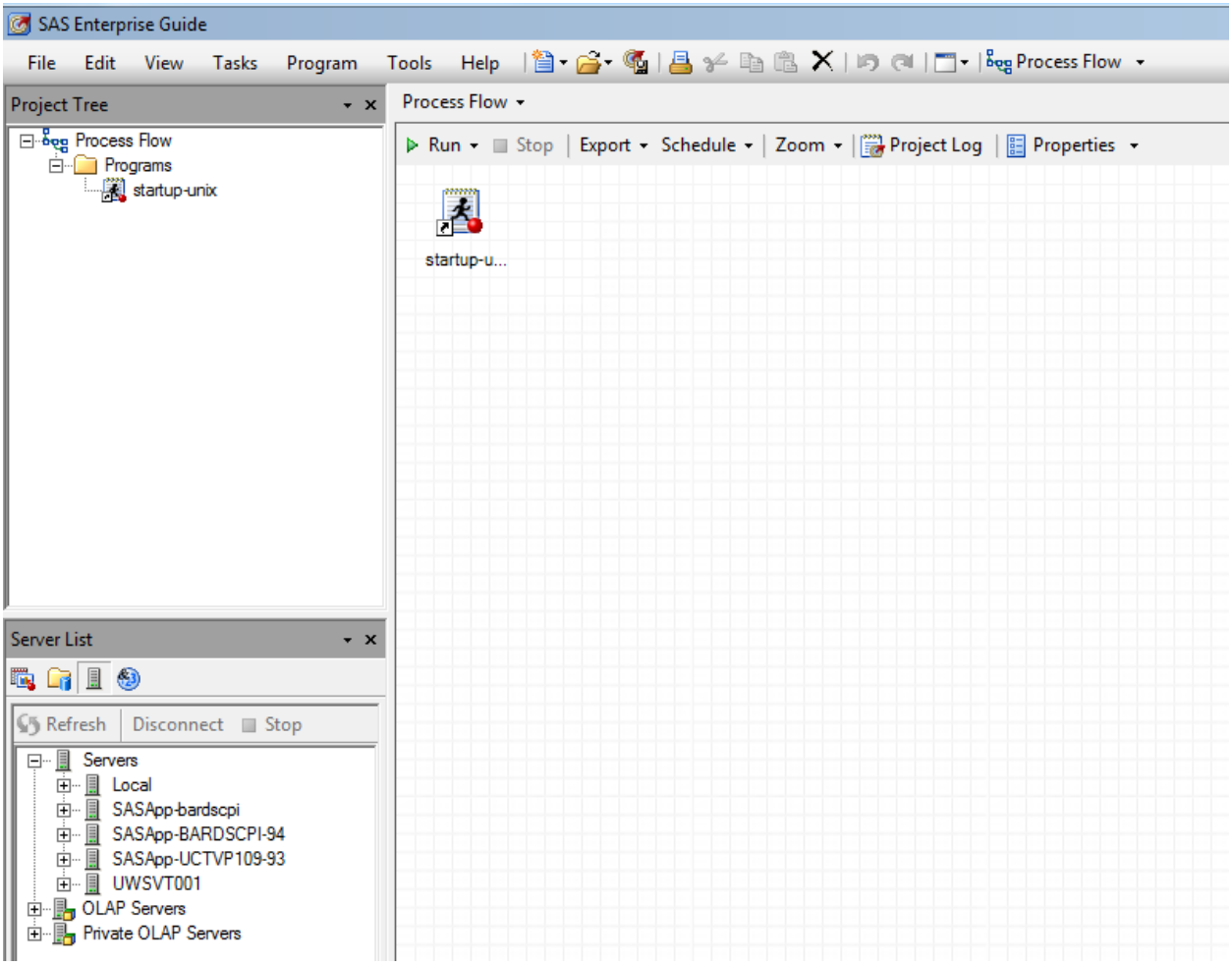
To save the file on a UNIX server, first click Servers, and a list of available servers will appear. Double clicking it will open your UNIX home directory, and then you can navigate down to the directory of interest.

Using Enterprise Guide to Edit a .sas File on UNIX:

Editing SAS code is just as easy. The File -> Open Program dialog box allows you to navigate the remote directory structure, the same way as the Save dialog box. After you open a program, a link is added to the process flow. After that point, to edit the program, just double-click the link.

Executing Code on UNIX:

To execute the code, just hit ‘Run’. As long as your Enterprise Guide session is connected to the UNIX SAS Server, the code will be running there. You can also highlight a piece of code and run it. The log, the lst and the output will be displayed in separate windows. Running SAS job on UNIX is as easy as running it on PC SAS.



Conclusion

When working across PC SAS and UNIX SAS, we want to ensure the usability of the same software/program in different environments. Those environments could be diverse which is important to appreciate as a SAS developer. Good programming practice can play an important role in this process while Enterprise Guide provides an effective and efficient alternative.

Reference

PC SAS Programmer Facing UNIX? SAS Enterprise Guide to the Rescue. Quentin McMullen, http://bi-notes.com/2013/03/sas-enterprise-guide-unix-putty/

SAS Programs from Unix to PC and Beyond. Christopher Kania, http://www.lexjansen.com/pharmasug/2006/TechnicalTechniques/TT09.pdf