Paper 3196-2015

Tips for Managing SAS® Work Libraries

Thomas E. Billings, MUFG Union Bank, N.A., San Francisco, California
Avinash Kalwani, Oklahoma State University, Stillwater, Oklahoma

ABSTRACT

The Work library is at the core of most SAS® programs, but programmers tend to ignore it unless something breaks. This paper first discusses the USER= system option for saving the Work files in a directory. Then, we cover a similar macro-controlled method for saving the files in your Work library, and the interaction of this method with OPTIONS NOREPLACE and the syntax check options. A number of SAS system options that help you to manage Work libraries are discussed: WORK=, WORKINIT, WORKTERM; these options might be restricted by SAS Administrators. Additional considerations in managing Work libraries are discussed: handling large files, file compression, programming style, and macro-controlled deletion of redundant files in SAS® Enterprise Guide®.

INTRODUCTION

In the most common usage, the SAS® system automatically assigns a Work library for each SAS session, and programmers write and read intermediate files they create in the Work library during that session. For the most part, programmers take the Work library for granted, i.e., pay very little attention to it unless they run out of space or the job is running very slowly.

The Work library is at the core of most SAS jobs, and it should be managed when appropriate. The SAS system provides a number of system options related to Work libraries:

1. USER=
2. WORK=
3. WORKTERM, NOWORKTERM
4. WORKINIT, NOWORKINIT.

Options 2-4 above may be restricted by your local SAS Admins, but if not restricted they can be useful in certain contexts. Besides these options, there are other system/data set options that are relevant in this context, i.e., COMPRESS=.

This paper is intended for programmers who need to manage their Work libraries:

- when you run out of space in the Work library,
- for debug, audit, or other analysis, you would like to access the contents of the Work library *after* the job has run or even while running “live”, from another SAS session,
- if you are writing large intermediate files and the job runs slowly.

We begin with the basics: simple ways to keep a copy of your Work library and the pros and cons of this approach.

SYSTEM OPTION USER= OR
REPLACE REFERENCES TO WORK WITH A MACRO VARIABLE, &MYLIB

Changing the location of the Work Library via the USER= system option

One method is to write all the one-level data set names to a permanent library by using the USER= system option; this option cannot be restricted by SAS Admins. Sample code for this is as follows.

```
libname test "pathname";
options user=test;

data company;
   input id $ dept $;
   datalines;
X Sales
Y Acct
Z HR
;
```

The ‘company’ data set will be stored in the permanent library as specified in LIBNAME ‘test’.

Replace references to WORK with a macro variable, &mylib

Instead of using one-level data set names in your SAS code like, say, file “myfile” in the code below:

```
proc sort data=myfile nodupkey;
    by key1 key2;
run;
```

you can write the code as:

```
proc sort data=&mylib..myfile nodupkey;
    by key1 key2;
run;
```

where &mylib is a %GLOBAL macro variable that resolves to any of the following:

- WORK – i.e., the standard Work library allocated by the SAS system,
- a meta-LIBNAME for which you have write access,
- your locally-defined LIBNAME for which you have write access.

In this approach, 2-level names are explicitly used to reference files in DATA and PROC steps, with the LIBNAME portion being a global macro variable. By setting the macro variable to a LIBNAME other than WORK, you can save - in a permanent library - the files that would otherwise be created in a temporary Work library and then deleted at the end of the job.

Note that you can use &mylib selectively, i.e., for only some of the files created in a job, leaving redundant or minor/unimportant files in the standard Work directory. Also, you can define &mylib1, &mylib2, etc., and have multiple libraries (you might want to do that to support different logical layers in a large program). Of course, it is also possible to dynamically change the directory that USER= points to in a program, to support multiple libraries.

At some sites, the Work library is assigned to the fastest disk drives available, with permanent files assigned to slower disks. The Work library may also have a (much) larger disk space allocation than most individual users; this is especially common in university computers.   These factors may impact the feasibility of the &mylib and USER= methods described above.

INTERACTIONS: THE &MYLIB METHOD VS. OPTIONS NOREPLACE AND SYNTAX CHECK MODE

When OPTIONS NOREPLACE is specified then permanent data sets with the same name will not be replaced. Syntax check mode – specified via OPTIONS SYNTAXCHECK or OPTIONS DMSSYNCHK – sets OPTIONS OBS=0 NOREPLACE when it encounters an error in a DATA or PROC step that creates a data set. This puts the SAS system into a special scan mode where your program statements are checked for syntax errors but data sets are not replaced. Syntax check has a number of exceptions and might not function exactly as you expect; see Billings (2013, 2014) for details. The SAS system documentation advises that NOREPLACE does **not** apply to data sets in the Work library.

The end result of this is that there can be differences in which data sets are created/saved if you use the &mylib method with NOREPLACE and/or the syntax check options set NOREPLACE after an error. (Note that the NOREPLACE option is not commonly used by itself, but the syntax check options are widely used.) If you select these options then depending on your code and possibly the presence of an error, some data sets that are downstream would be created if &mylib resolves to WORK, but not created when &mylib points to another LIBNAME. The code/programs below illustrate this point, and include some extra code (i.e., the code to force an error) discussed later.

Test program #1 sets &mylib to WORK and tests the effect of an error coupled with NOREPLACE and NOSYNTAXCHECK.

```
%let mylib=WORK;   *  test #1:  WORK library;
options mprint nocenter;
options nosyntaxcheck replace;    * make sure OPTIONS are set as desired;
data &mylib..temp;    * create test data set;
      x=1;
      output;
      stop;
run;

options noreplace;
data &mylib..mistake;   * force an error;
      y = 1;
      z = fakefunction(y);
      stop;
run;

data &mylib..temp;   * check if replaced;
      set &mylib..temp;
      y = 1;
run;

proc print data=&mylib..temp;
      title "test #1 - using WORK libname";
run;
```

The log shows that the WORK.temp file is indeed replaced despite the use of NOREPLACE:

```
34         data &mylib..temp;   * check if replaced;
35          set &mylib..temp;
36          y = 1;
37         run;

NOTE: There were 1 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.TEMP has 1 observations and 2 variables.
```

and the final version of WORK.temp is indeed the updated version as modified in the code above:

Obs	x	y
1	1	1

Test program #2 sets &mylib to a meta-LIBNAME and runs code similar to program #1.

```
%let mylib=CDMCNTL;      *  test #2:  permanent library;
%*libname &mylib. "path/address_here";     * not needed as this
                                             is a meta-libname;
options nosyntaxcheck replace;    * make sure OPTIONS are set as desired;
data &mylib..temp2;
      x=1;
      output;
      stop;
run;

options noreplace;
data &mylib..mistake2;
      y = 1;
      z = fakefunction(y);
      stop;
run;

data &mylib..temp2;
      set &mylib..temp2;
      y = 1;
run;

proc print data=&mylib..temp2;
      title "test #2 - using permanent libname";
run;
```

this case, the log shows that the update of file temp2 is **not** done due to NOREPLACE:

```
34         data &mylib..temp2;
35          set &mylib..temp2;
36          y = 1;
37         run;

NOTE: There were 1 observations read from the data set CDMCNTL.TEMP2.
NOTE: The data set CDMCNTL.TEMP2 has 1 observations and 2 variables.
WARNING: Data set CDMCNTL.TEMP2 was not replaced because of NOREPLACE option.
```

and per the PROC PRINT, file temp2 is unchanged, i.e., it does not include variable y:

Obs	x
1	1

As an exercise, readers are encouraged to take the code for the 2 test cases above, copy/paste to their systems, modify &mylib and LIBNAME as needed, and set the SYNTAXCHECK (or DMSSYNCHK) options and explore how the code works under syntax check mode. (Both test cases include extra code - a forced error - for that specific purpose.) If you are running in SAS® Enterprise Guide®, each test case should be encapsulated in a separate Program Task; for the reasons, see Billings (2014).

Pros and cons of the & mylib method

Pros:

- Flexible - lets you save select files that would otherwise be deleted from the Work library. This provides you with a copy of intermediate files after the run, and can be useful for debug, audit, and/or regulatory needs. More flexible than the USER= option.
- Can use regular Work library during development and switch to saving files before migrating the job to production.
- Under your control; cannot be restricted by SAS Admins.

Cons:

- Space limits -- some users have small disk allocations; not practical for extremely large files.
- Source code is slightly more complex.
- Using this method for select files can cause confusion/errors if you end up with 2 versions of a file in a single job, i.e., a WORK version and a &mylib version with the same file name.

USING THE PATHNAME FUNCTION TO GET THE WORK LIBRARY ADDRESS:

The PATHNAME function returns the physical location (path) for any library using a designated libref. You can get the full physical address of the Work library using the macro system function %SYSFUNC:

```
%let address = %sysfunc(pathname(WORK));
%put &address;
```

If you have the (system-assigned) Work library address and save it using some of the options discussed in the next section, then you will be able to access the files after the job has completed. Additionally, you can use this approach to get, say, the address of your Work library in a SAS® Data Integration Studio® job, and use it in a LIBNAME statement (but not one named WORK) in SAS Enterprise Guide to access, “live” or simultaneously, the Work library for your SAS Data Integration Studio job(s). This is very handy for debug and analyses, as the reporting tools of SAS Data Integration Studio are very weak compared to those in SAS Enterprise Guide. Note that for this access method to work, you will probably need to be signed in as the same userid in both SAS sessions.

In standard runs, the SAS system creates a Work directory for each SAS session. In most setups, the name of the Work directory has some randomness, making the actual address of that directory unknown unless you use the PATHNAME function and/or use the system options described in the next section. This prevents data contention issues and provides a limited degree of “security by obscurity”.

SAS SYSTEM OPTIONS WORK=, WORKINIT, AND WORKTERM

The SAS system options WORK=, WORKINIT, and WORKTERM may play an important role when you work with temporary Work libraries. These options are valid only at SAS invocation – via the "sas" command to your operating system – or in the system configuration file. This limits these options primarily to batch runs, and there is a further relevant constraint – the use of these options can be restricted by SAS Admins.

WORK= option

This option provides an alternative to deleting the Work data sets, i.e., it lets you save those files to another SAS library. You can also use the USER= system option to store temporary data sets in the USER library rather than in the WORK library.

WORKTERM option

This option specifies whether or not SAS erases the files present in the Work library at the termination of a SAS session. This is the default option if you don't need to keep the files at the end of the SAS session.

NOWORKTERM option

This option prevents the deletion of the data sets present in the Work library, but it does not affect the initialization of the Work library. By default SAS initializes the Work library whenever a SAS session starts, deleting the pre-existing data sets.

WORKINIT option

This is a SAS invocation system option, which controls whether the Work data library is initialized at SAS session invocation. It deletes files that exist from a previous SAS session in an existing Work library.

NOWORKINIT option

This option saves/prevents erasure of the data sets that exist from the previous SAS session in the Work library at SAS session invocation. However, you must specify NOWORKTERM in the previous SAS session in order to reuse the Work directory that you saved, else the data sets in the Work library will already be erased before the run per the previous WORKTERM option.

Comparing WORKTERM, WORKINIT and the other methods

The WORKINIT system option initializes the Work data library and deletes all data sets at SAS invocation. The WORKTERM system option controls whether SAS deletes Work data sets when you end the SAS session. Nowadays, many users invoke SAS via point-and-click icons, and they don't directly invoke SAS via their operating system command line. Also, most users don't have the privileges required to update the system configuration files to use WORK=. The result is that the WORK=, WORKTERM, WORKINIT options are useful primarily for batch (production) runs, where allowed by local SAS Admins. Meanwhile, most of us who are not working in batch will find the USER= and/or &mylib methods to be the easiest way to save files in the Work library.

ADDITIONAL CONSIDERATIONS IN MANAGING WORK LIBRARIES

Additional considerations in managing Work libraries include the following.

Writing large Work files and file compression

- If you are writing very large Work files, then deleting files when they are no longer needed may reduce your job's run time and free up space in your disk allocation. Use PROC DATASETS and/or PROC DELETE to remove files that are no longer needed.
- If your Work libraries are large, limit the number of versions you save to avoid wasting disk space. The other users on your system will thank you.
- Use data set options or system options: COMPRESS=YES, CHAR, or BINARY, when appropriate for the files you are writing. This increases CPU time but decreases I/O cycles, and most SAS jobs are I/O-bound.
- COMPRESS=YES or CHAR is most effective for data that are predominantly character variables (in terms of space per record), with BINARY most effective for data that are predominantly numeric.

Programming style: use VIEWs when possible

- Some SAS programs are written as a sequence of small DATA and PROC steps, each of which performs only a few processes on target data sets. Often, multiple small DATA steps can be combined into a single DATA step for efficiency. The statements/data set options: KEEP, KEEP=, DROP, DROP=, RENAME, RENAME=, WHERE, WHERE= are frequently useful in this context.
- DATA step VIEWs and PROC SQL views are tools that every programmer should be familiar with. For example, consider source code that consists of multiple DATA steps operating on a file, which is then input to PROC SORT to create the target output file. In some cases, it may be possible to encapsulate the transformations from multiple DATA steps into a single DATA step VIEW, which is used as input to the PROC SORT. In this scenario, the VIEW runs and applies the transformations in a temporary data set which is then sorted. The temporary data set is automatically deleted by the SAS system after the sorting is completed; see Billings (2007) for more information.

Work files in SAS Enterprise Guide process flows: to delete or not to delete?

- If you are working with code in Program Tasks in SAS Enterprise Guide that produce a large number of Work files, then when the Program Task completes, the resulting Process Flow page may contain a large number of files and file icons: so many files that it can be difficult to find/pick out the files of interest for further analyses.
- The deletion of redundant/unneeded Work files can be placed under macro control. That is, write a macro that deletes redundant Work files at the end of the Program Task code. The deletion is controlled by a %GLOBAL macro variable that is set at the top of the project or process flow. This approach can be used in batch or windowing environments, but it is most useful in SAS Enterprise Guide.
- Macro-controlled deletion of intermediate files can be a useful debug tool; it is not limited to concerns re: cluttered Process Flow pages.

SUMMARY:

1. The system option USER= and macro-based &mylib methods provide simple ways to save the files in your Work library. These methods cannot be restricted by SAS Admins; they also have interactions with/are constrained by the syntax check options and OPTIONS NOREPLACE.
2. If they are allowed at your site (by SAS Admins), the system options WORK=, WORKINIT, WORKTERM when used appropriately can let you save and reuse files in your Work library that would otherwise be deleted when your SAS session terminates. These options are most useful for batch programs.
3. When appropriate, use file compression options and limit the Work files you save to those needed for debug/audit, to make efficient use of disk space on your system and as a courtesy to other users.
4. Programming style – combining DATA steps and using VIEWs – and deleting unneeded Work files in some environments can make your programs more efficient and easier to maintain/work with.

REFERENCES

Note: all URLs quoted or cited herein were accessed in March 2014.

Billings T. (2007) Enhance Your Programming with SAS® DATA Step Views. *WUSS Conference Proceedings,* 2007. URL: http://www.lexjansen.com/wuss/2007/ApplicationsDevelopment/APP_Billings_EnhanceProgramming.pdf

Billings T. (2013) An Overview of Syntax Check Mode and Why it is Important . *WUSS Conference Proceedings,* 2013. URL: http://wuss.org/Proceedings13/12_Paper.pdf

Billings T. (2014) Differences in Functionality of Error Handling Features, SAS® Enterprise Guide® vs. Batch. *WUSS Conference Proceedings,* 2014. URL: http://www.wuss.org/proceedings14/13_Final_Paper_PDF.pdf

SAS Institute, Inc. SA*S(R) 9.3 System Options: Reference, Second Edition.* Online documentation:

- USER=
  http://support.sas.com/documentation/cdl/en/lesysoptsref/64892/HTML/default/viewer.htm#p1tl8adik7ypwun1utdwxaauf9wq.htm
- WORK=
  http://support.sas.com/documentation/cdl/en/lesysoptsref/64892/HTML/default/viewer.htm#p1er6tm8fay8u2n1fhktmeoy2be4.htm
- WORKINIT
  http://support.sas.com/documentation/cdl/en/lesysoptsref/64892/HTML/default/viewer.htm#p09alrx4a9rop4n13xu3jpu37v3y.htm
- WORKTERM
  http://support.sas.com/documentation/cdl/en/lesysoptsref/64892/HTML/default/viewer.htm#p182lylgccdsgin14sgkqd4heyiz.htm

CONTACT INFORMATION:

Thomas E. Billings
MUFG Union Bank, N.A.
Credit Strategies Group
350 California St.; 9th floor
MC H-925
San Francisco, CA 94104

Phone: 415-273-2522
Email: tebillings@yahoo.com