Let SAS® Do Your DIRty Work

Richann Watson

ABSTRACT

Making sure that you have saved all the necessary information to replicate a deliverable can be a cumbersome task. You want to make sure that all the raw data sets and all the derived data sets, whether they are Study Data Tabulation Model (SDTM) data sets or Analysis Data Model (ADaM) data sets, are saved. You prefer that the date/time stamps are preserved. Not only do you need the data sets, you also need to keep a copy of all programs that were used to produce the deliverable, as well as the corresponding logs from when the programs were executed. Any other information that was needed to produce the necessary outputs also needs to be saved. You must do all of this for each deliverable, and it can be easy to overlook a step or some key information. Most people do this process manually. It can be a time-consuming process, so why not let SAS® do the work for you?

INTRODUCTION

Making a copy of all information needed to produce a deliverable is time-consuming and is normally done using

- Windows® Explorer: copy and paste to a new folder for archiving
- Command interpreter (CMD shell)*: copy or move commands

Both of these methods require you to create an archive directory and either copy or move the files manually. Most people go the route of using Windows Explorer due to familiarity. This can tie up your computer while the files are being archived. CMD is faster and commands can be set to run in the background if you know how to access CMD and know the necessary commands. But this still requires a manual approach. However, knowing a few basic CMD commands, you can use SAS to execute these commands and do a lot of directory maintenance for you.

* UNIX® if SAS is in a UNIX environment

BASIC CMD COMMANDS

Table 1 is a list of basic CMD/UNIX commands that can be used to do directory maintenance.

CMD Command	UNIX Command	Description
dir ***dir***	ll ***dir***	List all the files and directories under ***dir*** **
cd ***dir***	cd ***dir***	change directory to ***dir***
mkdir ***dir*** or md ***dir***	mkdir ***dir***	create a directory ***dir***
del ***file***	rm ***file***	delete ***file***
copy ***file1 file2***	cp ***file1 file2***	copy ***file1*** to ***file2*** **
move ***file1 file2***	mv ***file1 file2***	rename or move ***file1*** to ***file2***

Table 1. Basic CMD/Unix Commands

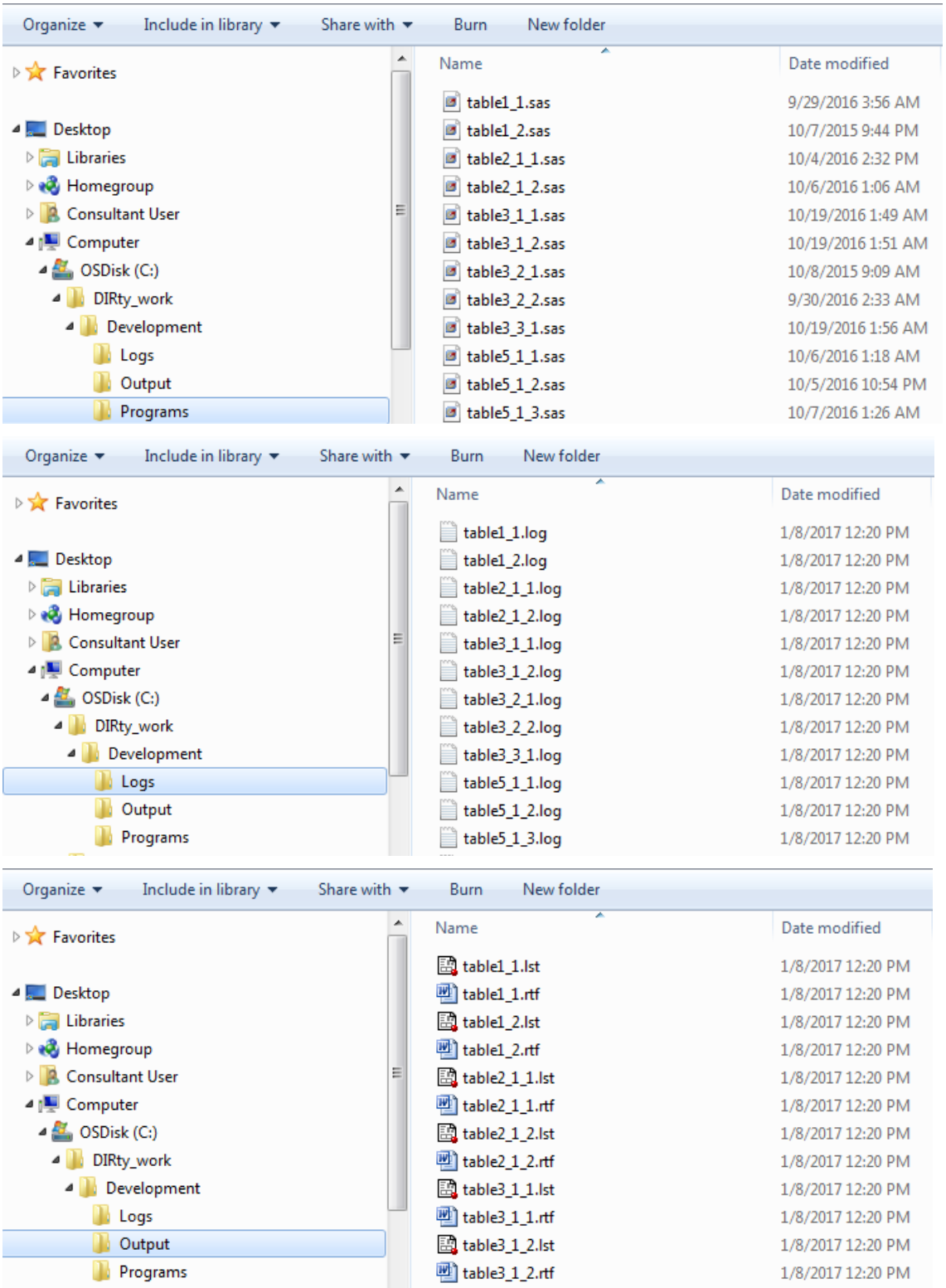
Note that CMD/UNIX commands are case sensitive.

When referencing ‘dir’ in the filename and pipe syntax below for UNIX ‘dir’ would be changed to ‘ll’.

HYPOTHETICAL SITUATION

You have several programs with a specific naming convention that are executed on a monthly schedule. You need to archive the programs, logs and outputs for those programs after each execution. This can be done by manually creating an archive folder and then copying only the necessary files. This manual process would need to be done each time the programs are executed. It can get to be a cumbersome task and it can be easily forgotten. However, this can be automated by using SAS to ‘talk’ to CMD or UNIX.

The current directory structure has a Development area with three subfolders for Programs, Logs and Output (see Display 1). Ideally the necessary files in the subfolders need to be copied to an archive folder that has yet to be created.



Display 1. Files in Working Directory

CREATING THE ARCHIVE FOLDER STRUCTURE**

When archiving the necessary files, it is normally desirable to have the archive folder named with a date to easily identify when the archive occurred and to set up the subfolders in the same structure as the main folder. Rather than creating this new folder and its subfolders via Windows Explorer or CMD, SAS can create an archive folder that is automatically named based on the date of creation.

** Note: For illustration purposes, CMD.exe (command line interpreter) was used. The commands for CMD are similar to UNIX with some minor differences which are noted in Table 1 above.

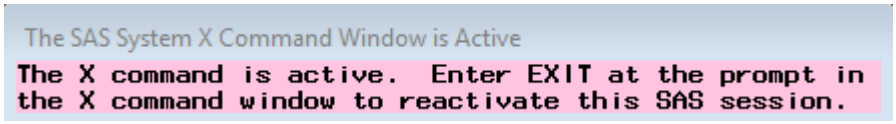
1. Create a macro variable that will be used to create the archive folder that has the date embedded:

```
/* create a macro variable that is date dependent */
data _null_;
  call symputx('archive', 'archive_' || lowcase(strip(put(today(),
yymmddn8.))));
run;
```

2. Set the following options:

```
/* XWAIT - user must enter EXIT after command to get back to SAS session */
/* XSYNC - SAS waits for other applications to finish before returning   */
options noxwait xsync;
```

If xwait is turned on then the message shown in Display 2 will appear after the execution of each x command.



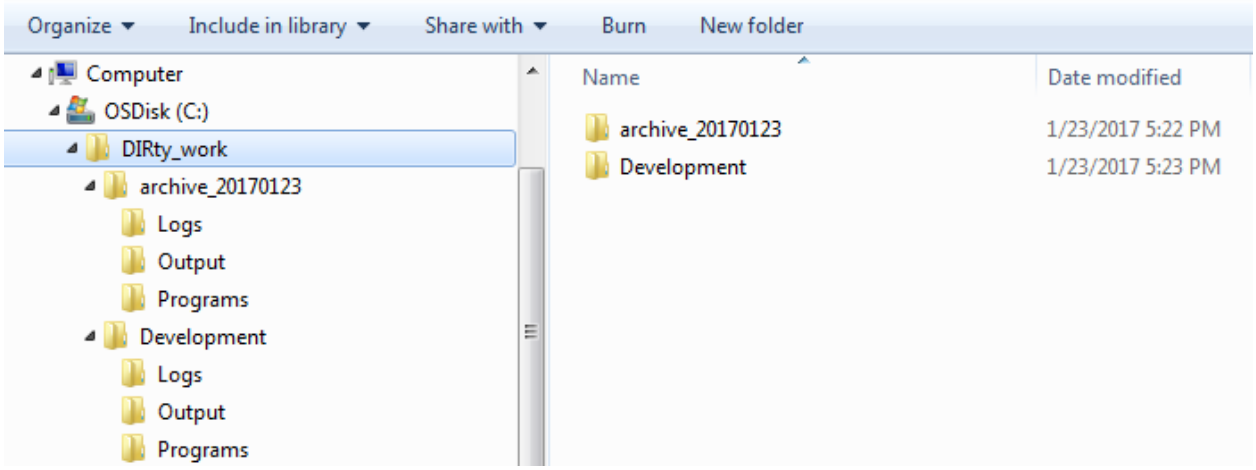
Display 2. XWAIT Command Message

3. Issue a series of CMD commands that will create the archive directory structure:

```
/* point to the main folder where the working directory is stored */
x cd c:/DIRty_work;

/* create directory to archive data using archive macro variable created */
x mkdir &archive;

/* note that you are currently at main directory need go to &archive */
/* change directories to point to the new archive directory          */
x cd &archive;

/* create sub directories under &archive to match main directory */
x mkdir Logs Output Programs;
```

The x command allows the user to submit CMD commands without leaving a SAS session.

The above commands will create an archive folder structure that mimics the 'Development' folder structure (see Display 3).



Display 3. Archive Folder Structure

RETRIEVE THE FILES TO BE ARCHIVED

Based on this hypothetical situation, we know the programs that are being run monthly will have a specific naming convention. The filename can be used to retrieve the programs that are being executed monthly. We kept the date of each program in addition to keeping the filename in case it is needed for future reference.

1. Using CMD you can read in the list of all files that are in a specified directory and create a SAS data set as illustrated in Table 2.

```
/* read all file name & properties in the directory to a file 'source' */
/* 'Pipe' is a way to communicate between processes (SAS and Windows). */
/* It allows for reading/writing between applications.                  */
/* This example, SAS will be reading from Windows                       */
filename source pipe      'dir C:\DIRty_work\Development\Programs';

/* It will create a SAS data set from the file that was created using */
/* the filename and pipe syntax                                       */
data pgm;
  infile source lrecl=32727 truncover scanover;
  input dirtext $200.;
run;
```

dirtext
Volume in drive C is OSDisk
Volume Serial Number is FAC6-8D8A
Directory of C:\DIRty_work\Development\Programs
01/08/2017 12:24 PM <DIR> .
01/08/2017 12:24 PM <DIR> ..
09/29/2016 02:56 AM 9,959 table1_1.sas
10/07/2015 08:44 PM 9,129 table1_2.sas
10/04/2016 01:32 PM 11,865 table2_1_1.sas
10/06/2016 12:06 AM 8,079 table2_1_2.sas
10/19/2016 12:49 AM 9,838 table3_1_1.sas
10/19/2016 12:51 AM 10,959 table3_1_2.sas
10/08/2015 08:09 AM 10,440 table3_2_1.sas
09/30/2016 01:32 AM 7,610 table3_2_2.sas
10/19/2016 12:56 AM 10,141 table3_3_1.sas
10/06/2016 12:18 AM 7,674 table5_1_1.sas
10/05/2016 09:54 PM 6,303 table5_1_2.sas
10/07/2016 12:26 AM 6,348 table5_1_3.sas
10/06/2016 12:16 AM 6,555 table5_2_1.sas
10/06/2016 12:17 AM 6,600 table5_2_2.sas
14 File(s) 121,500 bytes
2 Dir(s) 127,699,636,224 bytes free

Table 2. Records in pgm SAS data set

Note that the information provided may be slightly different depending if you are using CMD or UNIX.

2. The data set as is contains one variable per record with file property information. In addition it contains a variety of records that are not needed. We are only interested in SAS programs with a naming convention of 'table3'. Using the file properties that were retrieved in the 'pipe' can help to subset to the necessary records. The data in Table 2 shows that the SAS program name is in the 5th token in the string for each record that has '.sas' and the date is the first token when a space is used as the delimiter. Using the scan function, the SAS program and the date of the file can be retrieved and be used to subset the data accordingly. Table 3 demonstrates the use of filename to subset the data to only keep files that have 'table3' in the filename. If there are spaces in the file name then retrieving the filename will require extracting each token of the filename individually and then concatenating the tokens to create the full filename.

```
/* only interested in SAS programs (i.e. '.sas' extensions) and */
/* files with a specific naming convention                      */
data pgm (where=(fname ? 'table3') );
  infile source lrecl=32727 truncover scanover;
  input dirtext $200.;
  length fname $20.;
  format fdt date9.;

  /* subset for only SAS programs */
  if index(lowcase(dirtext), '.sas');

  /* determine the name of the SAS program */
  fname = tranwrd(scan(dirtext, 5, ' '), '.sas', '');

  /* determine the date/time stamp */
  fdt = input((scan(dirtext, 1, ' ')), mmddyy10.);
run;
```

dirtext	fname	fdt
10/19/2016 12:49 AM 9,838 table3_1_1.sas	table3_1_1	19OCT2016
10/19/2016 12:51 AM 10,959 table3_1_2.sas	table3_1_2	19OCT2016
10/08/2015 08:09 AM 10,440 table3_2_1.sas	table3_2_1	08OCT2015
09/30/2016 01:32 AM 7,610 table3_2_2.sas	table3_2_2	30SEP2016
10/19/2016 12:56 AM 10,141 table3_3_1.sas	table3_3_1	19OCT2016

Table 3. Records in pgm SAS data set subset for SAS programs with current date

Note that in Table 3**Error! Reference source not found.**, the 'fname' does not contain the '.sas' extension. This is purposely left off since the log and corresponding output are more than likely going to have the same filename as the SAS program. Therefore, rather than have to either replace '.sas' with the necessary extension or strip off the '.sas' extension later, it is left off.

3. Using the subset data, a macro variable can be created to loop through each of the subfolders in the main directory and copy the necessary files over to the archive folders:

```
/* create a macro variable of SAS program names concatenated into one */
proc sql noprint;
  select fname into :saspgm separated by ", "
  from pgm;
quit;
```

For this example, macro variable &saspgm will contain the following value "table3_1_1, table3_1_2, table_3_2_1, table_3_2_2, table3_3_1".

The macro variable that contains the list of SAS programs can be used to loop through each of the subfolders in the main directory and copy to the corresponding folder in the archive directory:

```
/* use macro variable to loop through sub folders in main directory    */
/* retrieve SAS programs, logs and outputs and copy to archive folders */
%macro archive;

  /* change directories to the main directory */
  x cd c:\DIRty_work;

  %let x = 1;
  %let pgmscn = %scan(%bquote(&saspgm), &x, '._', nakq);

  %do %while (&pgmscn ne );

    x copy Development\Programs\&pgmscn..sas     &archive.\Programs\.;
    x copy Development\Logs\&pgmscn..log         &archive.\Logs\.;
    x copy Development\Output\&pgmscn..rtf       &archive.\Output\.;

    %let x = %eval(&x + 1);
    %let pgmscn = %scan(%bquote(&saspgm), &x, '._', nakq);

  %end;
%mend archive;
%archive
```

An alternate method to using a macro call is to use CALL EXECUTE within a data step. CALL EXECUTE will produce SAS statements and then execute the SAS statements at the step boundary (e.g. RUN;):

```
data _null;
  set pgm;
  call execute('x cd c:\DIRty_work;');

  call execute(cats("x 'copy Development\Programs\", strip(fname),
                    ".sas &archive.\Programs\.';"));
  call execute(cats("x 'copy Development\Logs\", strip(fname),
                    ".log &archive.\Logs\.';"));
  call execute(cats("x 'copy Development\Output\", strip(fname),
                    ".rtf &archive.\Output\.';"));
run;
```

FURTHER POSSIBILITIES

This process can be made more robust by looking at each program and scanning for SAS data sets and/or other types of inputs used in the program so that those items can be copied as well:

```
/* read the SAS program into a file     */
/* create a SAS data set of the program */
/* each line of code is one record      */
filename code&x "c:\DIRty_work\Development\Programs\&pgmscn..sas";
data pgm&x;
  infile code&x missover length=reclen;
  input line $varying800. reclen;


  /* look for records with the specified libname name */
  dsloc = prxmatch("/RAW|SDTM|ADAM/i", line);

  /* keep records where RAW, SDTM or ADAM libname is found */
  if dsloc > 0;
run;
```

Within the above DATA step, you can use a variety of SAS functions to scan through each line of code and extract the names of the data sets used. Those data set names can then be stored in a macro variable so that they can be copied using the above approach.

For a more detailed look at how this can be done see the following paper “SAS® Macro Tool to Find Source Data Sets Used in Programs” http://www.lexjansen.com/mwsug/2012/PH/MWSUG-2012-PH05.pdf

For a more detailed explanation on using PRXMATCH to search a text string refer to the following paper “Perl Regular Expressions in SAS® 9.1+ - Practical Applications” http://www.lexjansen.com/pharmasug/2012/TA/PharmaSUG-2012-TA08.pdf

CONCLUSION

Let SAS and CMD/UNIX talk to each other and do the grunt work for you. This will save you time and frustration. In addition, it will allow you to automate a process that can be easily overlooked.

REFERENCES

SAS Support, http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#npipes.htm

UNIX commands, http://www.shareittips.com/data/unix.png

CMD commands, http://ss64.com/nt/

ACKNOWLEDGMENTS

Thanks to Nancy Brucken and Deanna Schreiber-Gregory for reviewing the paper and providing additional insight.