PharmaSUG 2013 - Paper TF06

Let SAS® Do Your DIRty Work

Richann Watson, Experis, Batavia, OH

ABSTRACT

Making sure you have all the necessary information to replicate a deliverable saved can be a cumbersome task. You want to make sure that all the raw data sets are saved, all the derived data sets, whether they are SDTM or ADaM data sets, are saved and 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 when the programs were executed. Any other information that was needed to produce the necessary outputs needs to be saved. All of these needs to be done for each deliverable and it can be easy to overlook a step or some key information. Most people do this process manually and 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

The following 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

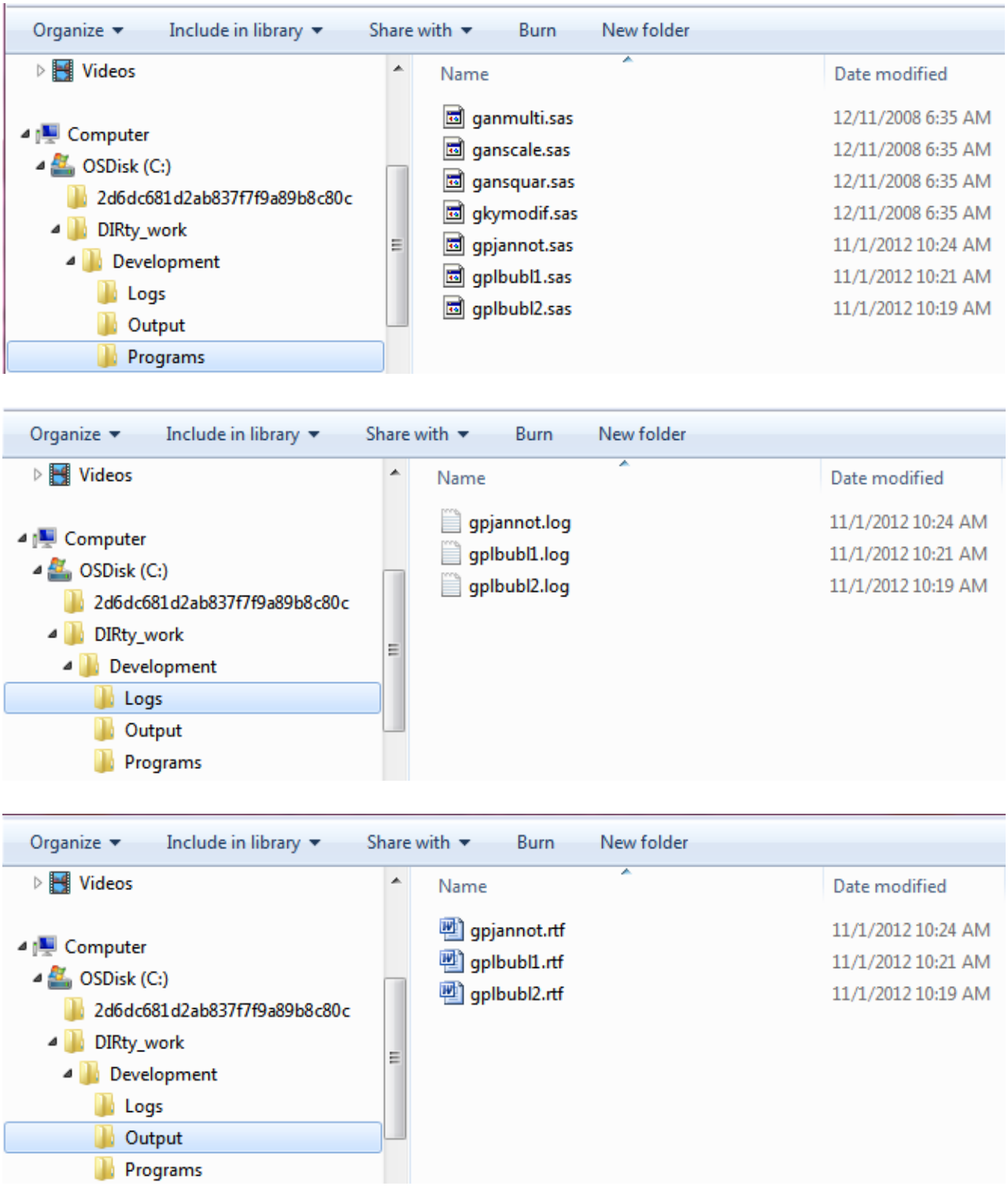
Note that CMD/UNIX commands are case sensitive.

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

HYPOTHETICAL SITUATION

You have several programs that are updated and 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 in 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 above.

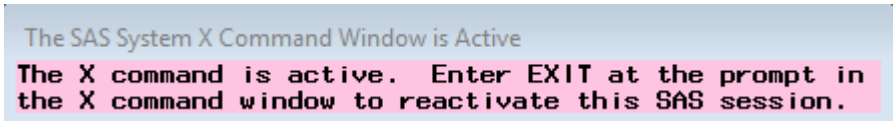
1. Create a macro variable that will be used to create the archive folder that has the date embedded.

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

If xwait is turned on then the following message will appear after the execution of each x command.



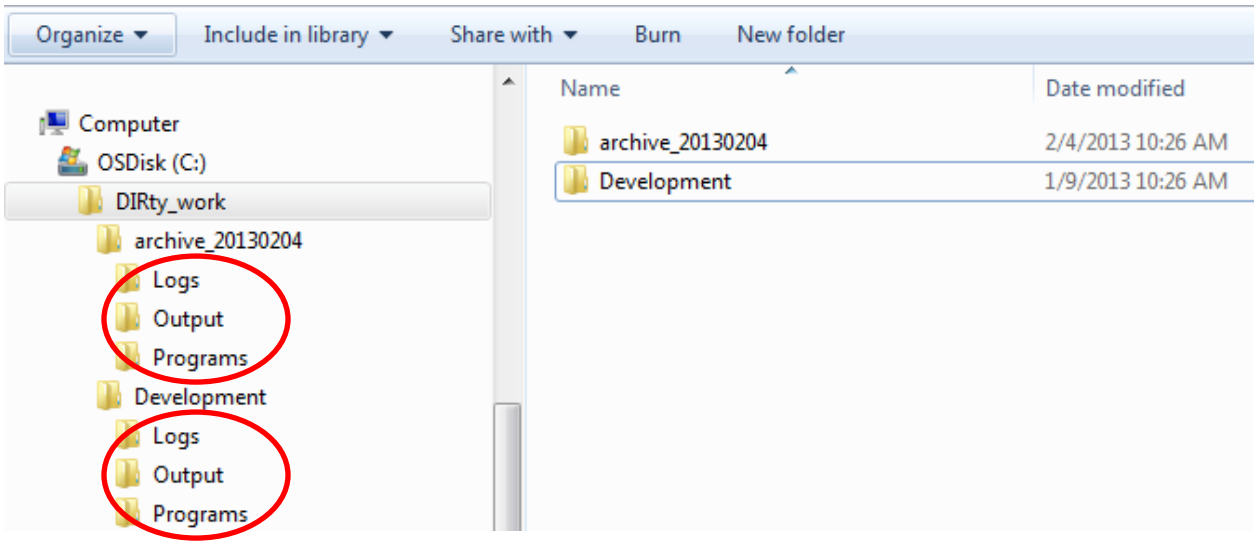
Display 2. XWAIT Command Message

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 a directory that will archive data using macro variable created above */
x mkdir &archive;

/* change directories to point to the new archive directory just created      */
/* note that you are currently at c:/DIRty_work so to go to the sub directory */
/* &archive you only need to specify cd &archive.  However if you were at a   */
/* different directory level then you can specify "cd c:\DIRty_work\&archive" */
x cd &archive;

/* create sub directories under the archive directory 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 the hypothetical situation, we know the programs that are to be run monthly will have to be updated monthly as well. The date can be used to retrieve the programs that are updated monthly.

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

```
/* read all file name & properties in the specified 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 94AC-D4DF
Directory of C:\DIRty work\Development\Programs
11/01/2012 09:30 AM <DIR> .
11/01/2012 09:30 AM <DIR> ..
12/11/2008 06:35 AM 2,699 ganmulti.sas
12/11/2008 06:35 AM 3,053 ganscale.sas
12/11/2008 06:35 AM 2,615 gansquar.sas
12/11/2008 06:35 AM 2,134 gkymodif.sas
11/01/2012 09:24 AM 3,066 gpjannot.sas
11/01/2012 09:21 AM 1,833 gplbubl1.sas
11/01/2012 09:19 AM 1,662 gplbubl2.sas
7 File(s) 17,062 bytes
2 Dir(s) 264,030,380,032 bytes free

Table 1. Records in pgm SAS data set

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

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 and programs that have been recently modified. Using the file properties that were retrieved in the 'pipe' can help to subset to the necessary records. The data in Table 1 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.

```
/* only interested in SAS programs (i.e. '.sas' extensions) and files with a*/
/* specific date so the date needs to be extracted                          */
data pgm (where=(fdt=today()) );
  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 */
  /* want to strip off the '.sas' portion  */
  fname = tranwrd(scan(dirtext, 5, ' '), '.sas', '');

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

dirtext	***fname***	***fdt***
11/01/2012 09:24 AM 3,066 gpjannot.sas	gpjannot	1-Nov-12
11/01/2012 09:21 AM 1,833 gplbubl1.sas	gplbubl1	1-Nov-12
11/01/2012 09:19 AM 1,662 gplbubl2.sas	gplbubl2	1-Nov-12

Table 2. Records in pgm SAS data set subset for SAS programs with current date

Note that the 'fname' does not contain the '.sas' extension. This is purposely left off since the log and corresponding output is 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 and just the actual filename is kept.

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 the 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 "gpjannot, gplbubl1, gplbubl2".

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 each sub folders in main directory      */
/* to retrieve SAS programs, logs and outputs and copy to the 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.mwsug.org/proceedings/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.pharmasug.org/proceedings/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

http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#pipeover.htm

UNIX commands

http://www.shareittips.com/data/unix.png

CMD commands

http://ss64.com/nt/

ACKNOWLEDGMENTS

Thanks to Nancy Brucken for reviewing the paper and providing additional insight.

CONTACT INFORMATION

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

Richann Watson
Experis
richann.watson@experis.com