Paper 026-2021

A SAS® Toolbox for File and Folder Manipulation: Copy, Rename, Delete, or Zip via Functions or X Commands

Aaron Brown, Cognia, Inc.

ABSTRACT

This paper discusses various utilities within SAS® for manipulating folders and files within a Windows environment, including creating folders, copying files, renaming files, deleting files and folders, and zipping folders. It includes examples of using tools like X commands, the DLCREATEDIR option, the macro language, and functions like FDELETE and FCOPY. We first discuss several tools, then show two sample projects that utilize them.

INTRODUCTION

Within the past year, I had a handful of projects that required me to do things such as copying, renaming, or zipping files from one location to another. While it could have easily been done manually, such manual labor would have been very tedious and time-consuming. So, instead, I turned to SAS®. I was unable to easily find sample code online to do exactly what I wanted, but I did find several useful examples of parts of what I needed. The purpose of this paper is to consolidate those tools into one toolkit, with the hope of this paper adding to the online resources already available, so that the next person has less places they need to delve. See the Recommended Reading at the end of this paper for a list of websites along with a short description of what tools they exemplify.

We will begin by going through several different tools. I will describe how each works and give at least a small example of SAS code. Then, I will walk through two projects that serve as examples of how to utilize these tools.

TOOL #1: X COMMANDS

X commands, also known as shell commands, allow the user to tell SAS to execute system commands. In a Windows environment, this is essentially letting you issue DOS commands.

For these to work, you need to have X commands enabled in your SAS environment. You can check this by using the OPTIONS procedure, as shown below:

```
PROC OPTIONS OPTION=XCMD; RUN;
```

If the output (in the log) states NOXCMD, you probably need to ask your SAS administrator to alter the value; unlike most SAS options, XCMD is valid only at startup during a the SAS session. Some companies restrict the use of X commands, in which case you will need to find an alternative tool.

Since X commands are not available to everyone, I will limit the toolset shown here to piping. Piping enables you to search a folder structure (including subdirectories) for any files that fit a proclaimed syntax.

For example, the code below

```
OPTIONS NOXWAIT;
X "cd &dir";
X "dir  *.rtf  /b /s > files.txt";
```

does the following:

1. The NOXWAIT option tells SAS not to prompt you for every change you make via X commands.
2. The first X command sets the current directory (cd) to a file path saved to the macro variable *dir*.
3. The second X command "pipes" everything in the current directory (dir) that has .rtf in its name to a file in the current directory named files.txt. The "/b" option suppresses the writing of metadata to the text file. The "/s" option includes the full filepath of the files. (If you want just the file names, not the entire file path, exclude "/s".)

We will use piping to create a list of the files we want to manipulate. We can then read that into SAS via a DATA step like the below

```
DATA MyListOfFiles;
  INFILE "&dir\files.txt" LRECL=256 PAD MISSOVER;
  INPUT myZip $ 1-256;
RUN;
```

X commands can do a lot more than piping. Some further examples are shown in the recommended reading. For example, you can delete a folder via the following command, where FolderToDelete is a macro variable with the full file path.

```
X RD /S /Q "&FolderToDelete";
```

Later, I will show how to use FDELETE to delete files and folders. A benefit of X commands over FDELETE is that an X command can delete a folder that has contents, whereas FDELETE only works on empty folders.

TOOL #2: USING FILE PATHS IN DATA STEPS TO CREATE NEW FILE PATHS

This tool is more of a concept than an exact tool. After we have read our file paths into SAS via the piping and the first tool, we can then manipulate the data in a DATA step to get the results we need. If we are copying those files to a new location, we should build the file path for the new location. If we are renaming the files, we should build the new name. Renaming is a rather simple example, so I'll use it below.

Let's say we piped the locations of all the files we want to rename. Each file is named something like "1001_Files.zip". We want to rename it to something like "School Reports.zip". Below is a simple data step that can do that by first extracting the ID number (e.g., 1001) then using that to create the new file extension.

Here, we set the current directory to where we wanted and we piped without including the file path. That means we know that the ID number is the first 4 characters of our file name. The folder structure is set and dependent on the ID number, so we include it while building the new file paths.

```
X "cd &dir";
X "dir  *.zip /b  > files.txt";
DATA MyListOfFiles;

INFILE "&dir\files.txt" LRECL=256 PAD MISSOVER;
INPUT myZip $ 1-256;
LENGTH ID $4. oldName newName $256.;
ID=substr(myZip,1,4);

oldName="&dir\"||ID||'\'||ID||"000\"||myZip;
newName="&dir\"||ID||'\'||ID||"000\School Reports.zip";
RUN;
```

TOOL #3: RENAMING FILES

Since we just did the set-up to rename a file, let's go ahead and show that tool. We simply use the RENAME function.

```
DATA MyListOfFiles;

SET MyListOfFiles;
rc=RENAME(oldName,newName,'FILE');

RUN;
```

The variable rc is populated with a numeric value. If the file renaming was successful, it has a value of 0; if there was a failure or error, it returns another value. Checking this variable via the PRINT or FREQ procedures can help you check if everything was renamed correctly.

TOOL #4: MAKING FOLDERS: THE DLCREATEDIR OPTION AND LIBREFS

By default, SAS does not create folders. However, you can 'trick' SAS into creating folders by using the OPTION DLCREATEDIR and library references. DLCREATEDIR tells SAS that, if you create a library reference for a folder that does not exist, it should make that folder. However, a significant limitation of this option is that it can only create the lowest folder in a hierarchy. That is, if you want to create O:\MyFiles\File5, you need O:\MyFiles to exist first. If it does not already exist, you have to create it before you can create O:\MyFiles\File5.

And, as expected, you need a folder to exist before you can move files into it.

If you know the exact folder structure you want to create, you can create it via a hard-coded manner in which you create any higher-level folders first, followed by sub-folders. On the other hand, if the folder structure changes, you need to dynamically create the folders in order. We will go over both methods in the example projects.

Once you have a folder you need to create saved to a character variable, you can use that to create the folder. You can do this by assigning the folder to the library reference via the CALL EXECUTE command; by creating the library reference, you also create the folder. You can use a second CALL EXECUTE command to clear the library reference.

For example, if we are looping through a dataset with the variable *folderName* holding the entire file path of folders we want created, we simply these two lines in our DATA step in order to create the folders:

```
CALL EXECUTE("libname temp '"||STRIP(string)||"';");
CALL EXECUTE("libname temp clear;");
```

Caution: you can try to tell SAS to create a zipped folder by naming a library reference with the ending ".zip". However, it is not truly a zipped folder even if Windows sometimes recognizes it as such. We will discuss later how to truly zip a folder.

TOOL #5: COPYING FILES

Once you have created the folders you want to copy a file to, you need to copy the actual file. To do this, first assign the file you want copied (with the full file path) and where it want it copied to (with the full file path and the file's name and extension) to two different file references.

In the example below, we have the original file saved as a macro variable *filenames* and the new location saved as a macro variable *moveTo.* Thus, we can copy the folder via these lines of code:

```
FILENAME ref1 "&filenames";
FILENAME ref2 "&moveTo";
%LET rc = %SYSFUNC( FCOPY( ref1, ref2 ));
```

Like RENAME, FDELETE returns a value of 0 if there is no error. If you want to check if an error occurred, this line of code prints one to the SAS log.[1]

```
DATA _null_;
      IF &rc NE 0 THEN %PUT %SYSFUNC( SYSMSG() );;
RUN;
```

You can also use FCOPY within a DATA step, like we used the RENAME function. This example is simply showing a different way to call the function, suing the %SYSFUNC utility.

---

[1] The two semi-colons are required. One ends the %PUT statement, while the other ends the IF-THEN statement.

TOOL #6: DELETING A FILE, AND CHECKING IF A FILE EXISTS

If you want to delete an individual file – not a folder, but a file – you can do so with the FDELETE function. (You can also delete folders with this function, but I find that a bit trickier; the next tool will cover deleting folders.)

As an example, let's say we want to delete that text file we piped at the beginning of our program. This can be done via the following code:

```
FILENAME temp "&dir\files.txt";
DATA _null_;
      IF (FEXIST('temp')) THEN rc=FDELETE('temp');
RUN;
FILENAME temp CLEAR;
```

First, we assign the file we want to delete to a filer reference. Then the DATA step uses the FEXIST function checks if the file exists, returning a 1 if it does and a 0 if it does not. Since SAS evaluates non-zeroes as true, if the file exists, it then executes the FDELETE function on the file reference. Following best practices, we then clear the file reference.

SAS will treat a zipped folder as a file for these purposes.

TOOLS #7 & 8: MACROS TO ZIP FOLDERS AND DELETE FOLDERS

Generally, when I am programming I prefer not to lift code as-is from the internet. Instead, I like to get an understanding of how the code works and incorporate it into my programs. For the task of zipping folders as well as the task of deleting a folder without X commands (along with any sub-folders and files within it), I was not able to do this.

Instead, I found a very helpful macro called *SASZip_Lite* that goes over how to zip folders. (There are easier ways if you have a software like WinZIP or 7-Zip installed on your computer, but this macro utilizes SAS, Visual Basic, and the default zipping capability of a Windows environment.) Read the SAS Global paper for full details on all its utilities.

For deleting folders, I found advice on a SAS community post, which I copied and put within a macro. The big problem that this macro solves is that FDELETE can only delete an empty folder. Thus, if you want to simply tell SAS "delete this folder", your code has to look inside the folder, delete everything inside it, then delete the folder. The macro does not work if the folder itself has sub-folders, though; for that, X commands are better (or one could presumably recursively call the macro to clear out and delete sub-folders, but I settled for an X command.)

The source code for these functions is in the recommended reading. A copy is also in the project code. [2]

There is also a FILENAME ZIP utility in SAS 9.4, which can manipulate .zip or .gz zipped folders. For more details, see the link to Chris Hemedinger's blog in the recommended reading.

TOOLS #9: FOR DIFFERENTLY-STRUCTURED FOLDERS, LOOPING THROUGH FOLDER HIERARCHIES

Under tool 4, we noted that you have to create a parent folder before you can create a sub-folder. This is easy to do if the folder structure is known to you, but harder if it can change. In one of our projects, we are copying the file structure from one location to another; thus, we do not know ahead of time what the structure is and it changes drastically folder to folder.

To solve this, I used Leonid Batkhan's advice regarding the n-th instance of a substring within a string to locate the index numbers of the slashes that demarcate folders. First, we create a function called

---

[2] The project code's version has two edits from the SAS Global paper. One is that the NOXWAIT comment has been suppressed as that option was already set. The other is that the *vbsdir* variable has been changed in order to work in my operating network. Set *vbsdir* to a folder which you have write and delete access to.

FINDNTH using the FCMP procedure. See this blog for the FCMP code.[3]

The code is as follows, and is run within a DATA step:

```
do j=1 to numFolders;
      p=findnth(moveDir,'\',j);
      string=substr(moveDir,1,p);
      if p NE 0 then do;
            CALL EXECUTE("libname temp '"||STRIP(string)||"';");
            CALL EXECUTE("libname temp clear;");
      end;
end;
```

where *numFolders* was defined earlier as the number of folders in the file path via

```
numFolders=count(moveDir,"\");
```

This loop executes once for each folder. The *findnth* function delivers the index value of the folder, then SUBSTR is used to truncate our file path to that folder. Tool 4 is then utilized to create the actual folder. Then the loop continues until there are not more folders (when p=0). Note: if the folder already exists, no harm is done; however, for the sake of efficiency, you could shift the j=1 to starting a higher number if you know some folders exist deep enough in. For example, in the project I was doing, I knew the four first folders already existed, so I could use *do j=4 to numFolders* instead.

SAMPLE PROJECTS

I learned the tools above by doing three or four different projects at work. To show them for this paper, I will first show a practical program that utilizes some of these tools. Then, I show an artificially convoluted program written to incorporate all of these tools.

SAMPLE PROGRAM

Let us assume we have an R-drive on our computer or network, and we want to copy files from a file path there (saved to a macro variable named &subfolder) to somewhere else (a folder location stored in &copyfolder). This program assumes all the files have different names.

To give this some extra utility, it will only copy files whose filename contains a character string stored in a macro variable &lookup. That is, this is a program to search and copy for select files.

First we pipe the contents of the R directory via X commands. The statement below

```
OPTIONS NOXWAIT;
%let dir=R:\&subfolder;
X "cd &dir";
X "echo off";
X "FOR /R . %F IN (*.*) DO echo %~tF %~zF %~F >> files.txt");
*line above gives a warning about macro resolution, but is not a
problem as long as %F does not exist;
X "echo on";
```

creates a text file with all the files in R, plus their metadata. Then, we read in the text file (reading each row as a long character string), then split out the data into different elements.

```
DATA MyFilesTest;
      INFILE "&dir\files.txt" LRECL=256 PAD MISSOVER;
      INPUT all_data $ 1-256;
RUN;
```

---

[3] https://blogs.sas.com/content/sgf/2019/06/26/finding-n-th-instance-of-a-substring-within-a-string/. For my purposes, I modified his code to use the FINDC function instead of FIND in PROC FCMP.

```
DATA MyFiles;
      SET MyFilesTest;
      CreationDate=input(substr(all_data,1,10),mmddyy10.);

      *isolate file size + file path & name;
      restOfStuff=substr(all_data,21);
      FileSizeChar=scan(restOfStuff,1,' ');
      FileSize=input(FileSizeChar,8.);
      FileNames=substr(restOfStuff,length(FileSizeChar)+1);

      FORMAT CreationDate date9.;
      DROP all_data restOfStuff FileSizeChar;
RUN;
```

Now, we prepare the data for copying.

```
DATA MyFiles;
      SET MyFiles;
      LENGTH myFile myPath myPath2 moveTo $256.;

      *separate the full file path & name into the path and the name;
      myFile=scan(filenames,-1,'\');
      myPath=substr(filenames,1,lengthn(filenames)-lengthn(myFile)-1);
      myPath2=STRIP("&CopyFolder");
      moveTo=STRIP(myPath2)||'\'||STRIP(myFile);
RUN;
```

Note: the CATS function might work better than the || operative in many cases, but I found it caused a buffering warning. Using || avoided that issue. Now, we flag the files we want to copy, based on a lookup. (You could modify this code to search via other means, such as the creation date or file size. If any of you want sample code that gives more search options, feel free to e-mail me. But I'm keeping it simpler for this example.)

```
DATA MyFiles2;
      SET MyFiles;
      IF FIND(UPCASE(myFile),UPCASE("&lookup")) THEN OUTPUT;
RUN;
```

And now we simply loop through the dataset to copy the files.

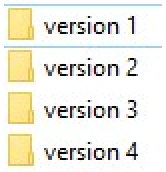
```
DATA MyFiles2;
      SET MyFiles2 END=lastobs;
      *get number of files to be moved;
      IF lastobs THEN CALL SYMPUTX('obsnum',_n_);
RUN;

*for each record you want to copy, copy the file;
%macro doThis;
%do i=1 %to &obsnum;
      data _null_;
            set MyFiles2(firstobs=&i obs=&i);
            CALL SYMPUTX('filenames',filenames);
            CALL SYMPUTX('moveTo',moveTo);
      run;
      filename ref1 "&filenames";
      filename ref2 "&moveTo";
      %let rc = %sysfunc( fcopy( ref1, ref2 ));
      data _null_;
            if &rc NE 0 then %put %sysfunc( sysmsg() );;
      run;
%end;
%mend;
%doThis;
```

Once this program runs, the files are copied.
The last step (below) is optional, but will delete the text file that was written out earlier.

```
*clean-up by deleting the text file created at the start;
filename myref "R:\&subfolder\files.txt";
DATA _null_;
      rc=FDELETE('myref');
RUN;
```

FULLER SAMPLE PROGRAM

Thie sample program is a bit more complicated and convoluted than such would probably be programmed in practice, but it was designed this way in order to maximize displaying our toolkit. We have a folder named “9999 FOLDERS”. It has subfolders labeled “version 1” through “version 4”, as shown below.



version 1
version 2
version 3
version 4

Graphic 1. Contents of 9999 Folders

The folder for version 2 has three sub-folders. The folder for version 3 is empty. Contents are each folder are shown in the table below.

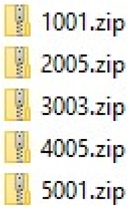
version 1	version 2/bad-DO NOT USE	version 2/original	version 2/patched	version 4
1001.sas 2005.sas 3003.sas 4005.sas 5001.sas	1001.sas 4005.sas 5001.sas	1001.sas 4005.sas 5001.sas	1001.sas 4005.sas 5001.sas	1001.sas

Table 1. Table Layout of 9999 Folders’ Content

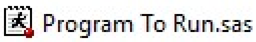
Each SAS program (that is, .sas file) is associated with an ID number. Our goal is to move the programs such that each ID number has its own zipped folder, containing the folder hierarchy. However, we do not want to include blank folders. We are also to exclude any ‘DO NOT USE’ programs.

Thus, for example, we should have a zipped folder named 1001 that contains a version 1, version 2/original, version 2/ patched, and version 4 folder and program. However, 2005 will only contain a version 1 folder. No ID number will have a version 3 folder as version 3 is blank.

Finally, we want to rename the programs. For simplicity sake, we will rename them to “Program to Run”. Our final output will be a set of five zipped folders. See screenshot below for them and the contents of 2005.

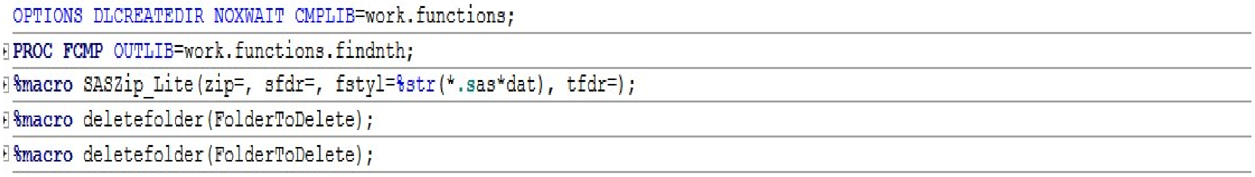


/2005.zip /2005.zip/version1

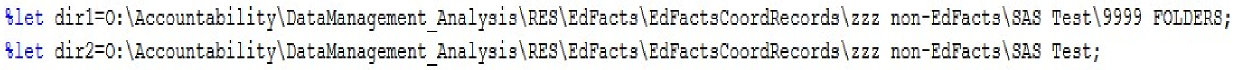


Graphic 2. Display of Desired Final Output, with Details on 2005.zip

We start by setting out options, macros, and directories. dir1 is where the files/folders to manipulate are stored and dir2 is where we want the new folders created. Full macros are in Appendix A.



```
OPTIONS DLCREATEDIR NOXWAIT CMPLIB=work.functions;
PROC FCMP OUTLIB=work.functions.findnth;
%macro SASZip_Lite(zip=, sfdr=, fstyl=%str(*.sas*dat), tfdr=);
%macro deletefolder(FolderToDelete);
%macro deletefolder(FolderToDelete);
```



```
%let dir1=O:\Accountability\DataManagement_Analysis\RES\EdFacts\EdFactsCoordRecords\zzz non-EdFacts\SAS Test\9999 FOLDERS;
%let dir2=O:\Accountability\DataManagement_Analysis\RES\EdFacts\EdFactsCoordRecords\zzz non-EdFacts\SAS Test;
```

Next we pipe all the SAS files within the “9999 FOLDERS” to a text file, then read in the text file and, using what we know of the file naming conventions, isolate what we need. We also have a line to delete any folders or files with DO NOT USE in their name.

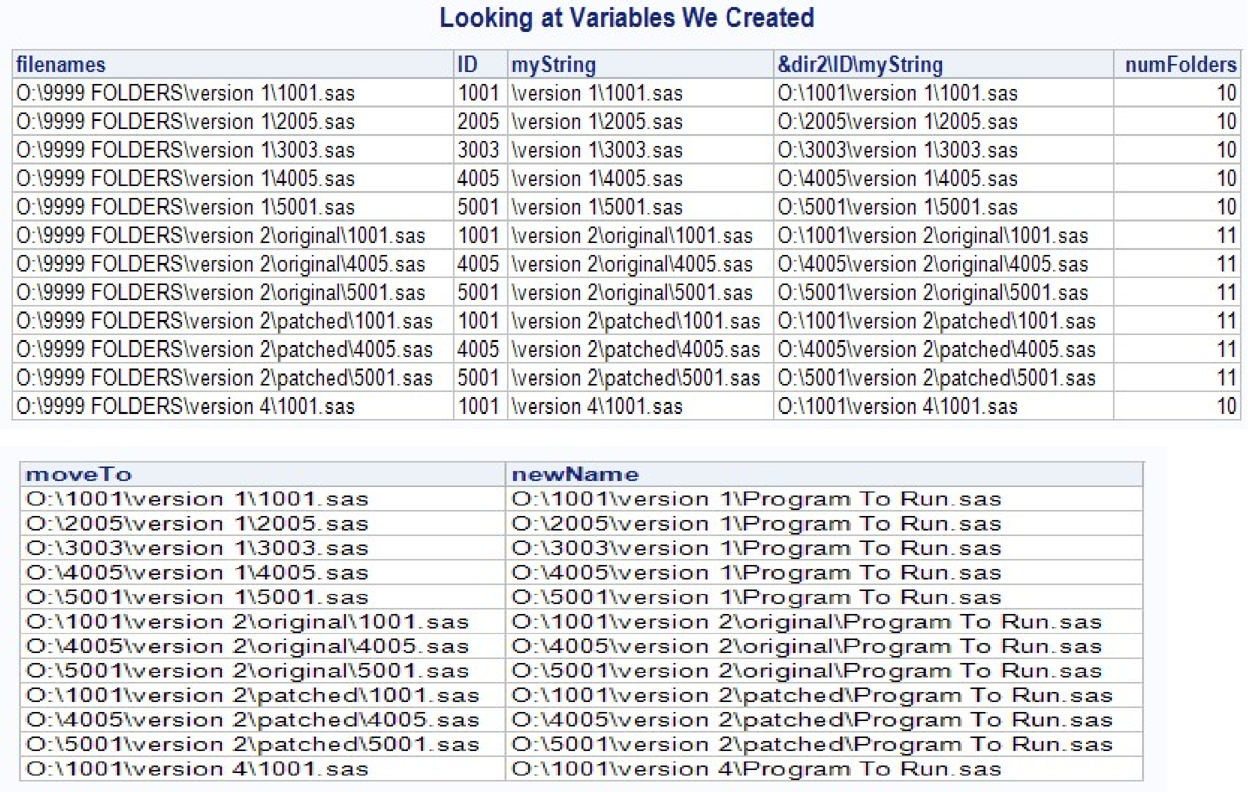
```
X "cd &dir1";
X "dir  *.sas  /b /s > files.txt";
*Read in the text file;
DATA MyFiles;
  INFILE 'files.txt' LRECL=256 PAD MISSOVER;
  INPUT filenames $ 1-256; *256 is still the max path length allowed
in dos/windows;

  *DELETE THE 'Do Not Use' FOLDERS;
  IF find(UPCASE(filenames),"DO NOT USE") THEN delete;

  length moveTo moveDir newName $256.;
  *CREATE DATA WE WILL NEED LATER ON ABOUT FILE STRUCTURE;
  *get ID number, extracting from the SAS files;
  ID=scan(filenames,-1,'\');
  ID=compress(ID,,'kd');
  *get the parts of filename we want to copy over;
  myString=substr(filenames,112);
  *create folder name || file name where should move files to;
  moveTo="&dir2.\"||STRIP(ID)||STRIP(myString);
  *create name of new file location without its file name;
  temp=findc(moveTo,'\',"i",-length(moveTo));
  moveDir=substr(moveTo,1,temp);
  *get number of folders;
  numFolders=count(moveDir,"\");
```

```
   *set up new name for file;
   newName=TRANWRD(moveTo,STRIP(ID)||'.sas','Program To Run.sas');
RUN;
```

We can check that we created the folder structure correctly by running a few PRINT procedures.

```
Title 'Looking at Variables We Created';
PROC PRINT DATA=MyFiles NOOBS LABEL;
      VAR filenames ID myString moveTo numFolders;
      LABEL moveTo='&dir2\ID\myString';
RUN;
PROC PRINT DATA=MyFiles NOOBS;
      VAR moveTo newName;
RUN;
```

Here is a subset of the PROC PRINT output. I have snipped the file path for readability. The first output lets us see how the original file name (filenames) was transformed into a new one (moveTo, there displayed via a label as how it was constructed.) The second output shows the original name and what it will be changed to when we rename the file.



Looking at Variables We Created

filenames	ID	myString	&dir2\ID\myString	numFolders
O:\9999 FOLDERS\version 1\1001.sas	1001	\version 1\1001.sas	O:\1001\version 1\1001.sas	10
O:\9999 FOLDERS\version 1\2005.sas	2005	\version 1\2005.sas	O:\2005\version 1\2005.sas	10
O:\9999 FOLDERS\version 1\3003.sas	3003	\version 1\3003.sas	O:\3003\version 1\3003.sas	10
O:\9999 FOLDERS\version 1\4005.sas	4005	\version 1\4005.sas	O:\4005\version 1\4005.sas	10
O:\9999 FOLDERS\version 1\5001.sas	5001	\version 1\5001.sas	O:\5001\version 1\5001.sas	10
O:\9999 FOLDERS\version 2\original\1001.sas	1001	\version 2\original\1001.sas	O:\1001\version 2\original\1001.sas	11
O:\9999 FOLDERS\version 2\original\4005.sas	4005	\version 2\original\4005.sas	O:\4005\version 2\original\4005.sas	11
O:\9999 FOLDERS\version 2\original\5001.sas	5001	\version 2\original\5001.sas	O:\5001\version 2\original\5001.sas	11
O:\9999 FOLDERS\version 2\patched\1001.sas	1001	\version 2\patched\1001.sas	O:\1001\version 2\patched\1001.sas	11
O:\9999 FOLDERS\version 2\patched\4005.sas	4005	\version 2\patched\4005.sas	O:\4005\version 2\patched\4005.sas	11
O:\9999 FOLDERS\version 2\patched\5001.sas	5001	\version 2\patched\5001.sas	O:\5001\version 2\patched\5001.sas	11
O:\9999 FOLDERS\version 4\1001.sas	1001	\version 4\1001.sas	O:\1001\version 4\1001.sas	10

moveTo	newName
O:\1001\version 1\1001.sas	O:\1001\version 1\Program To Run.sas
O:\2005\version 1\2005.sas	O:\2005\version 1\Program To Run.sas
O:\3003\version 1\3003.sas	O:\3003\version 1\Program To Run.sas
O:\4005\version 1\4005.sas	O:\4005\version 1\Program To Run.sas
O:\5001\version 1\5001.sas	O:\5001\version 1\Program To Run.sas
O:\1001\version 2\original\1001.sas	O:\1001\version 2\original\Program To Run.sas
O:\4005\version 2\original\4005.sas	O:\4005\version 2\original\Program To Run.sas
O:\5001\version 2\original\5001.sas	O:\5001\version 2\original\Program To Run.sas
O:\1001\version 2\patched\1001.sas	O:\1001\version 2\patched\Program To Run.sas
O:\4005\version 2\patched\4005.sas	O:\4005\version 2\patched\Program To Run.sas
O:\5001\version 2\patched\5001.sas	O:\5001\version 2\patched\Program To Run.sas
O:\1001\version 4\1001.sas	O:\1001\version 4\Program To Run.sas

Output 1. PROC PRINT Output of File Structure Creation

Now that we have created the metadata needed to make our folders, we set to creating them. We do this by looping through every file we need to move. The process shown below creates folders as needed via a CALL EXECUTE of a library reference, then moves the SAS program via a FCOPY call. Note the use of the *findnth* function to loop in order to create folders.

```
DATA _null_;
      SET MyFiles END=lastobs;
      IF lastobs THEN CALL SYMPUTX('obsnum',_n_);
run;
```

```
PROC SORT DATA=MyFiles; BY ID; RUN;
%macro doThis;
%do i=1 %to &obsnum;
      data temp;
            set MyFiles(firstobs=&i obs=&i);
            CALL SYMPUTX('filenames',filenames);
            CALL SYMPUTX('moveTo',moveTo);

            do j=7 to numFolders;
                  p=findnth(moveDir,'\',j);
                  string=substr(moveDir,1,p);
                  if p NE 0 then do;
                  CALL EXECUTE("libname temp '"||STRIP(string)||"';");
                  CALL EXECUTE("libname temp clear;");
                  end;
            end;
      run;
      filename ref1 "&filenames";
      filename ref2 "&moveTo";
      %let rc = %sysfunc( fcopy( ref1, ref2 ));
      data _null_;
            if &rc NE 0 then %put %sysfunc( sysmsg() );;
      run;
%end;
%mend;
%doThis;
```

The RENAME could have been incorporated into the code above, but for simplicity's sake I have separated it into one DATA step, as follows:

```
DATA _NULL_;
      SET MyFiles;
      rc=RENAME(moveTo,newName,'FILE');
run;
```

Now that we have created our folders, copied our files, and renamed our files, we need to zip them. (The 98+5 below comes from knowing that our new files are at the 98th character in the file path, then +5 for the slash and the 4-digit ID.)

```
proc sql;
      create table IDs as
      select distinct ID, substr(moveTo,1,%sysevalf(98+5)) AS zipDir
            /*98 is the position, +5 for "/XXXX"*/
      from MyFiles
      order by ID;
      title 'Folders to Zip';
      select * from IDs;
quit;
```

ID	zipDir
1001	O:\1001
2005	O:\2005
3003	O:\3003
4005	O:\4005
5001	O:\5001

Output 2. PROC SQL Output of Folders to Zip

We then implement a similar macro-loop in order to zip the folders. %SASZip_Lite is called to created a zipped version of the folder, then %deleteFolder (using an X command, since they have sub-folders) is called to delete the unzipped version. We also use FEXIST to check if the zipped folder already exists and to delete it if it does.

```
data _null_;
      set IDs end=lastobs;
      if lastobs then call symputx('obsnum',_n_);
run;
%macro doThis;
%do i=1 %to &obsnum;
      data _null_;
            set IDs(firstobs=&i obs=&i);
            CALL SYMPUTX('zipdir',zipdir);
            CALL SYMPUTX('fullzip',catx('.',zipdir,'zip'));
            CALL SYMPUTX('ID',ID);
      run;

      *if the zipped version already exists, delete it;
      filename temp "&fullzip";
      data _null_;
            if (fexist('temp')) then rc=fdelete('temp');
      run;
      filename temp clear;
      *create zipped version;
      %SASZip_Lite(zip=&zipDir,sfdr=&zipDir,fstyl=%str(*.*));
      *delete unzipped version;
      %deleteFolder(&zipDir);
%end;
%mend;
%doThis;
```

We have now accomplished our goal! As a final clean-up step, we can use FDELETE to delete the text file we created at the beginning of the program.

```
filename temp "&dir1\files.txt";
data _null_;
      if (fexist('temp')) then rc=fdelete('temp');
      PUTLOG rc;
run;
filename temp clear;
```

APPENDIX: FULL PROGRAM CODE FOR THE LONGER EXAMPLE

Some spacing may be odd due to copying this from SAS Enterprise Guide into Word. It is recommended that you copy this into SAS or a text editor for readability. This contains some comments that were omitted in the code excerpts above.

```
/*
This program does the following:

1. Pipes .sas files from H:\9999 FOLDERS to text
        Exclude any files in a DO NOT USE folder
2. Creates a folder for each individual ID
3. Moves the folder structure for each ID to its folder, copying then renaming SAS programs
4. Creates a zipped version of each ID's folder and deletes unzipped version
*/

OPTIONS DLCREATEDIR NOXWAIT CMPLIB=work.functions;
PROC FCMP OUTLIB=work.functions.findnth;
   FUNCTION findnth(str $, sub $, n);
      p = ifn(n>=0,0,length(str)+1);
      do i=1 to abs(n) until(p=0);
                *NOTE: edit from original: using FINDC instead of FIND;
         p = findc(str,sub,sign(n)*p+1);
      end;
      return (p);
   ENDSUB;
run;
%macro SASZip_Lite(zip=, sfdr=, fstyl=%str(*.sas*dat), tfdr=);
        /*****************************************************************
        The code posted below is provided "AS IS" with NO WARRANTIES.
        ZIP: directory and file name of zip archive
        SFDR: directory of source files (to be zipped)
        FSTYL: File type of source files; value: *.* as "zip a folder"
        TFDR: Target directory for unzipped files (for unzip)
        *****************************************************************/

        %local zip sfdr fstyl tfdr vbadir p q mode;
        /* Set up a temporary working folder for VBScript */
        %let vbsdir=O:\Accountability\DataManagement_Analysis\RES\EdFacts\EdFactsCoordRecords\zzz
non-EdFacts\testing;
        *options noxwait; *<--important option but already turned on for this entire program;
        /* To initiate a clean working space */
        %if %sysfunc(fileexist("&vbsdir"))=1 %then %sysexec rd /s/q "&vbsdir";
        %if %index(%upcase(&zip), .ZIP)=0 %then %let zip=&zip..zip;
        %let mode=;
        /* Compress (zip) files */
        %if %length(&sfdr)>0 and (%length(&zip)>0) %then %do;
        /* Extract directory name of the zip file, if no such folder, generate one */
        %let q=%sysfunc(tranwrd(&zip, %scan(&zip, -1, %str(\)), %str( )));
        %let q=%substr(&q, 1, %length(&q)-1);
        %if %sysfunc(fileexist("&q"))=0 %then %sysexec md "&q";
        /* Copy all requested files from a validated source folder to a temporary folder,
        and keep their original time stamps */
        %if %length(&sfdr)>0 and %sysfunc(fileexist("&sfdr"))=1 %then %do;
        %let mode=z;
        %sysexec md "&vbsdir";
        %if %qupcase(&fstyl)^=%str(*.*) %then %do;
        %sysexec md "&vbsdir.\temp_zip";
        %sysexec copy "&sfdr.\&fstyl" "&vbsdir.\temp_zip";
        %end;
        %end;
        %end;
        %else %if %length(&tfdr)>0 and %length(&zip)>0 and %sysfunc(fileexist("&zip"))>0
        %then %do; /* Unzip files */
        %let mode=u;
        %sysexec md "&vbsdir";
        %end;
        %if &mode=z or &mode=u %then %do;
        /* Generate VBScript based on different modes */
        data _null_;
        FILE "&vbsdir.\xpzip.vbs";
        put 'Set ZipArgs = WScript.Arguments';
        put 'InputFile = ZipArgs(0)';
        put 'TgtFile = ZipArgs(1)';
        put 'Set objShell = CreateObject("Shell.Application")';
        put 'Set source = objShell.NameSpace(InputFile).Items';
        put 'soucnt = objShell.NameSpace(InputFile).Items.Count';
        %if &mode=z %then %do;
        put 'CreateObject("Scripting.FileSystemObject").CreateTextFile(TgtFile,
        True).Write "PK" & Chr(5) & Chr(6) & String(18, Chr(0))';
        put 'objShell.NameSpace(TgtFile).CopyHere(source)';
        put 'Do Until objShell.NameSpace(TgtFile).Items.Count = soucnt';
        put 'wScript.Sleep 3000';
        put 'Loop';
        %end;
        %else put 'objShell.NameSpace(TgtFile).CopyHere(source)'; ;
        put 'wScript.Sleep 3000';
        run;
        /* Run VBScript file for data archiving */
        %if &mode=z %then %do;

        %if %qupcase(&fstyl)=%str(*.*) %then %sysexec CScript "&vbsdir.\xpzip.vbs"
        "&sfdr" "&zip";
        %else %sysexec CScript "&vbsdir.\xpzip.vbs" "&vbsdir.\temp_zip" "&zip";
        %end;
        %else %sysexec CScript "&vbsdir.\xpzip.vbs" "&zip" "&tfdr";
        %end;
        /* Clean up */
        %if %sysfunc(fileexist("&vbsdir"))=1 %then %sysexec rd /s/q "&vbsdir";
%mend SASZip_Lite;

*this version works if what you want to delete does not have subfolders;
*avoids using X commands;
%macro deletefolder(FolderToDelete);
        data work.FilesToDelete;
           length Name $ 100;
           keep Name;

           rc = filename("folder", "&folderToDelete.");
           dirId = dopen("folder");

           do i = 1 to dnum(dirID);
              Name = dread(dirId, i);
              output;
           end;

           rc = dclose(dirId);
        run;

        data _null_;
           set work.FilesToDelete end=lastDeleted;

           put "Deleting " Name;

           rc = filename("delfile", cats("&folderToDelete./", Name));
           rc = fdelete("delfile");
           put "del file " rc=;
           rc = filename("delfile");

           if lastDeleted then do;
              put "Deleting the folder '&folderToDelete.'";
              rc = filename("folder", "&folderToDelete.");
              rc = fdelete("folder");
              put "del folder " rc=;
              rc = filename("folder");
           end;
        run;
%mend;
*this version works regardless of if the folder has sub-folders;
*uses X commands;
%macro deletefolder(FolderToDelete);
```

```
    data _null_;
                X RD /S /Q "&FolderToDelete";
    run;
%mend;

%let dir1=O:\Accountability\DataManagement_Analysis\RES\EdFacts\EdFactsCoordRecords\zzz non-
EdFacts\SAS Test\9999 FOLDERS;
%let dir2=O:\Accountability\DataManagement_Analysis\RES\EdFacts\EdFactsCoordRecords\zzz non-
EdFacts\SAS Test;

*Pipe the location of all SAS programs to a text file;
X "cd &dir1";
X "dir  *.sas  /b /s > files.txt";
*Read in the text file;
DATA MyFiles;
  INFILE 'files.txt' LRECL=256 PAD MISSOVER;
  INPUT filenames $ 1-256; *256 is still the max path length allowed in dos/windows;

  *DELETE THE 'Do Not Use' FOLDERS;
  IF find(UPCASE(filenames),"DO NOT USE") THEN delete;

  length moveTo moveDir newName $256.;
  *CREATE DATA WE WILL NEED LATER ON ABOUT FILE STRUCTURE;
  *get ID number, extracting from the SAS files;
  ID=scan(filenames,-1,'\');
  ID=compress(ID,,'kd');
  *get the parts of filename we want to copy over;
  myString=substr(filenames,112);
  *create folder name || file name where should move files to;
  moveTo="&dir2.\"||STRIP(ID)||STRIP(myString);
  *create name of new file location without its file name;
  temp=findc(moveTo,'\',"i",-length(moveTo));
  moveDir=substr(moveTo,1,temp);
  *get number of folders;
  numFolders=count(moveDir,"\");
  *set up new name for file;
  newName=TRANWRD(moveTo,STRIP(ID)||'.sas','Program To Run.sas');
RUN;
Title 'Looking at Variables We Created';
PROC PRINT DATA=MyFiles NOOBS LABEL;
        VAR filenames ID myString moveTo numFolders;
        LABEL moveTo='&dir2\ID\myString';
RUN;
PROC PRINT DATA=MyFiles NOOBS;
        VAR moveTo newName;
RUN;

*Find each ID number and create individual folders, then copy files;
DATA _null_;
        SET MyFiles END=lastobs;
        IF lastobs THEN CALL SYMPUTX('obsnum',_n_);
run;
PROC SORT DATA=MyFiles; BY ID; RUN;
%macro doThis;
%do i=1 %to &obsnum;
        data temp;
                set MyFiles(firstobs=&i obs=&i);
                *create macro variables for copying file;
                CALL SYMPUTX('filenames',filenames);
                CALL SYMPUTX('moveTo',moveTo);

        *starting at 7 since the first 7 folders already exist, as shared between dir1 and dir2;
        *you could do j=1 if you want to be extra safe.  Since we use a libname statement, there
is no danger from assigning extra libraries;
                do j=7 to numFolders;
                        p=findnth(moveDir,'\',j);
                        string=substr(moveDir,1,p);
                        if p NE 0 then do;
                                CALL EXECUTE("libname temp '"||STRIP(string)||"';");
                                CALL EXECUTE("libname temp clear;");
                        end;
                end;
        run;
        filename ref1 "&filenames";
        filename ref2 "&moveTo";
        %let rc = %sysfunc( fcopy( ref1, ref2 )); *returns 0 if no error;
        data _null_; *if an error, print it;
                if &rc NE 0 then %put %sysfunc( sysmsg() );; *For some reason, needed two
semicolons to keep from erroring out;
        run;
%end;
%mend;
%doThis;

*Rename each file;
DATA _NULL_;
        SET MyFiles;
        rc=RENAME(moveTo,newName,'FILE');
run;

*Create zipped versions;
proc sql;
        create table IDs as
        select distinct ID, substr(moveTo,1,%sysevalf(98+5)) AS zipDir
                /*98 is the position, +5 for "/XXXX"*/
        from MyFiles
        order by ID
        ;
        title 'Folders to Zip';
        select * from IDs;
quit;
data _null_;
        set IDs end=lastobs;
        if lastobs then call symputx('obsnum',_n_);
run;
%macro doThis;
%do i=1 %to &obsnum;
        data _null_;
                set IDs(firstobs=&i obs=&i);
                CALL SYMPUTX('zipdir',zipdir);
                CALL SYMPUTX('fullzip',catx('.',zipdir,'zip'));
                CALL SYMPUTX('ID',ID);
        run;
        %put NOTE: for &i, &ID, zipdir is &zipdir and fullzip is &fullzip;

        *if the zipped version already exists, delete it;
        filename temp "&fullzip";
        data _null_;
                if (fexist('temp')) then rc=fdelete('temp');
        run;
        filename temp clear;
        *create zipped version;
        %SASZip_Lite(zip=&zipDir,sfdr=&zipDir,fstyl=%str(*.*));
        *delete unzipped version;
        %deleteFolder(&zipDir);
%end;
%mend;
%doThis;

*clean-up that text file made at start;
filename temp "&dir1\files.txt";
data _null_;
        if (fexist('temp')) then rc=fdelete('temp');
        PUTLOG rc;
run;
filename temp clear;
```

ACKNOWLEDGMENTS

The author thanks the SESUG chair and staff for this conference and the opportunity to present. And his former co-workers at the South Carolina Department of Education for gently pushing him into a project where these skills could be developed and for helping him refine the code into something practical.

REFERENCES

See the documentation behind Kai Koo's SASZip_Lite macro at *A SAS Macro to Zip and Unzip Files in MS Windows without Additional External Data Archiving Software* (SAS Global 2012): http://support.sas.com/resources/papers/proceedings12/057-2012.pdf

A discussion on what code to use to delete folders or files, including the source code used for the DeleteFolder macro, can be found at https://communities.sas.com/t5/SAS-Programming/delete-a-folder-directory-and-all-files-on-it/td-p/584870

For macros that copy, delete, or move without X commands, including good seed code for FDELTE, see Tom Bellmer's *Copy/Delete/Move files – without XCMD* http://www.sascommunity.org/planet/blog/category/sysexec/

For details on using a library reference to create folders, see Chris Hemedinger's *SAS trick: get the LIBNAME statement to create folders for you* at https://blogs.sas.com/content/sasdummy/2013/07/02/use-dlcreatedir-to-create-folders/
For information on FILENAME ZIP for .zip or .gz files, see his blogs at: https://blogs.sas.com/content/sasdummy/tag/filename-zip/
This is outside the scope of this paper, but if you want to put the output you are creating in a SAS program into a zipped folder, see this additional post by Chris Hemedinger: https://blogs.sas.com/content/sasdummy/2014/01/28/create-zip-ods-package/

Code for and discussion about Leonid Batkhan's *findnth* function can be found at his *Finding n-th instance of a substring within a string* at https://blogs.sas.com/content/sgf/2019/06/26/finding-n-th-instance-of-a-substring-within-a-string/

CONTACT INFORMATION

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

Aaron R. Brown
Cognia, Inc.
E-mail: aaron.brown@cognia.org